Technological Educational Institute of Crete

School of Applied Technology

Department of Electrical Engineering

Department of Mechanical Engineering

Remote Monitoring of the Bactrocera oleae(Gmelin) (Diptera: Tephritidae) Population Using an Automated McPhail Trap Controlled by a Raspberry Pi

by

KARAPATAKIS IAKOVOS

A dissertation submitted in partial fulfillment of the requirements for the degree of

Master of Science (MSc)

in

Advanced Production Systems, Automation and Robotics

June 2017

Heraklion – Greece

Student Name: Iakovos Karapatakis

SID: mth1

Supervisor: Associate Professor Dr. Lefteris Doitsidis,

Department of Electronic Engineering, Technological Educational Institute of Crete

I hereby declare that the work submitted is mine and that where I have made use of another's work, I have attributed the source(s) as stated in the bibliography.

June 2017

Heraklion – Greece

ABSTRACT

In the context of this thesis we have developed an autonomous electronic trap, which may be used for monitoring the population of Bactrocera oleae (Gmelin) (Diptera: Tephritidae) population in olive fields, so that it can help the end users (farmers and agriculturists) to identify potential threats on time and proceed in proper actions (bait sprays). The proposed system is based on the concept initially presented in (Doitsidis et al., 2016), but with some fundamental differences. All processes are performed on-board therefore minimizing the need to transfer large amount of information in a remote server and also the system has adequate computational power to support a variety of extra sensors (camera, temperature, humidity, barometric pressure, Ambient light {infrared, full-spectrum and human-visible light}, wind speed, battery voltage), which results in extended capabilities. The extra sensors provide additional information about the area of interested, therefore allowing the end users to take proper actions not only for bait sprays but also for other activities related to the olive tree day to day treatment. To support the functionality of the autonomous electronic trap detailed algorithms and procedures using machine vision techniques were implemented on-board and the insects were identified properly. A dedicated web server was also developed which can support the interaction of users with multiple devices, and allows them to monitor the data using a customized interface. The proposed system was tested and performed adequately for an extensive period of time.

Keywords

Remote insect monitoring; Raspberry Pi; McPhail trap; Bactrocera oleae (Gmelin); Olive tree; e-trap

ACKNOWLEDGEMENTS

This work was carried out from December 2016 to June 2017, and would not have been possible without the support and mentoring of my supervisor: Associate Professor Dr. Lefteris Doitsidis, and understanding of my colleagues and supervisor at work: Evgenios Karanikolaou, Giorgos Ioannou, Ilias Nitsos and Fotis Liotopoulos who put up with my chaotic workspace.

I would like to thank both Dimitris Pritsios and Lambros Bogdanis both for their helpful advice in the initial electronics design and for their help when I was trying to trouble shoot problems with the voltage divider and the GSM Modem interconnection.

Of course, I would like to show my gratitude to all the people who supported me during my undergraduate and graduate studies, professor Yiannis Bertachas, Georgios Tzanakis, professor Leonidas Naoumidis, Tassos Tsatsakis trusted and allowed me to use their equipment and offices to complete semester projects and study, Georgios Perakis, Antonis Stramatakis, Nikos Tsatsakis, Eva Strataki, Rena Katechaki, Evi Koutendaki and Michalis Panagiotakis and many more, have helped me out find accommodation, helped me get job interviews and gave me invaluable advice.. thank you all.

I also want to thank my parents Kostas and Ioanna Karapatakis and my grandmother Sophia Zambiaki for their support and encouragement throughout this and all my previous endeavors and for believing in me.

Finally, I would like to dedicate this thesis to my kids who are my inspiration and motivation. If there’s one thing to be learned from this, is that to succeed one has to believe in themselves and despite failure to keep on trying.

TABLE OF CONTENTS

LIST OF FIGURES

CODE LISTINGS

LIST OF ACRONIMS AND ABBREVIATIONS

Rpi Raspberry Pi, single board computer

OS Operating System

IIC, I2C, I^2C Inter-Integrated Circuit Bus

Inter-Integrated Circuit Bus is a two-wire bus that can be used to connect multiple devices that send and receive data

SD Secure Digital is a type of memory card used to store digital data. It was initially used for digital cameras.

MicroSD is an SD card with a smaller form factor that can fit into small devices like cell phones, tablets.

Swap file is a file on a disk or a whole partition that is used to temporarily store data that cannot fit into the main system RAM

RVD is a Resistive Voltage Divider, used to lower an input Voltage when two resistors are connected in series and we measure the voltage at the end of one of the two resistors. We can calculate and build RVD using any ratio we want taking into account that too small resistors may emit heat as well.

ADC is short for Analog to Digital Converter, it is a device that has the appropriate electronic components to take an analog signal as an input and convert it to a digital signal. In our case the ADC is given an analog voltage and through sampling (at specific time intervals) and performing quantization, it gives it a value that is between its predefined resolution steps.

PGA Programmable Gain Amplifier, used by the ADS1015 ADC to change the maximum voltage value that can be measured by the ADC. Changing the Gain can greatly increase the accuracy of our readings.

UART Universal Asynchronous Receiver/Transmitter, is a computer hardware device for asynchronous serial communication in which data format and transmission speeds are configurable.

1. INTRODUCTION

The early detection of pests relies on sampling, of different areas, in a field or a greenhouse that can be representative of the whole population. The McPhail trap is a device that is widely used in agriculture to monitor pest population growth over time. It falls into the general category of lure/bait traps because a water solution that attracts male subjects using a pheromone.

In the case of the olive tree, there are many insects that live off the stems and leaves of the olive tree but the insects that have the most economic impact on the yielded crop are the ones that attack the olive fruit. The most common in Greece is “Dakos”, *Bactrocera oleae (Gmelin) Dacus oleae (Diptera: Tephritidae).*

a

b c d e

Figure 1. a) Bactrocera oleae (Gmelin) Dacus (Diptera: Tephritidae) adult fly lays an egg in olive fruit. Photograph by Marshall W Johnson *(Johnson et al., 2011)* b) recent puncture on an olive, c) larvae in olive fruit, d) nymph (pupa) e) adult fly exit holes photographs by Giancarlo Dessì, Istituto Professionale Statale per l'Agricoltura e l'Ambiente "Cettolini" di Cagliari, distributed under a CC BY-SA 3.0 license.

The Bactrocera oleae (Gmelin), Dacus oleae (Diptera: Tephritidae) is a species of fruit fly that is phytophagous. The adult female flies affect the olive fruit of the olive tree, in which they lay fertilized eggs. The eggs grow into larvae (worms) that feed of the olive as they mature for a few weeks. After the larvae stage, they go in the nymph stage (pupa) where they cocoon for 10 days until they turn into adult flies. During the colder winter seasons nymphs, may stay in hibernation for up to 4 months.

Once the flies' biological cycle has been completed, the olive fruit left, are affected both quantitative and qualitative as they may: prematurely drop to the ground, have reduced mass or just produce olive oil of inferior quality (with greater acidity levels).

The incentive to develop an automated McPhail trap was to make objective, impartial and accurate measurements about the insect population that would lead to data-backed decisions about the optimal spraying period of the olive trees.

The first such systems were introduced using optoelectronics (Potamitis, Rigakis, & Fysarakis, 2014) and by machine vision (Fouskitakis, Doitsidis, Rigakis, & Sarantopoulos, 2015) (Doitsidis et al., 2016).

The main purpose of this thesis is to construct an automated McPhail trap using off the shelf electronics, a Raspberry Pi and a camera to remotely monitor the Bactrocera oleae (Gmelin) Dacus oleae (Diptera: Tephritidae) population and perform machine vision functions locally and send in reports via GSM/GPRS to a main server.

Complimentary to the traps primary objective, it is fitted with some environmental sensors that make it capable of monitoring weather parameters such as temperature, humidity, barometric pressure, wind speed, ambient light {infrared, full-spectrum and human-visible light}, and other system related parameters like battery voltage that was deemed critical for the autonomy of the device.

By recording and forwarding all this information to a remote server, we can recognize patterns, discover new links between environmental parameters and the population increase of the olive tree pests, alert the people responsible of spraying of the olive trees, and send personnel onsite to replace the battery or renew the water solution.

2. LITERATURE REVIEW

2.1. Lure / Bait trap usage

Pesticide misuse can cause health problems to people that come into direct contact with the chemicals (when spraying), even indirectly by consuming those produces due to bioaccumulation and biomagnification. On the other hand pesticide misuse can cause disruption to the environmental balance of the ecosystem when pests along with other insects and organisms develop resistance to chemicals due to overuse or are treated with wrong substances altogether (Norton, Rajotte, & Gapud, 1999).

For several decades' farmers use traps to catch, identify and count the number of pests to assess population growth and determine if it is the right time to spray their crops. For the olive fruit fly, the most commonly used McPhail trap liquid solution is a 2% water solution of ammonium sulfate (selective to Tephritidae) (Doitsidis et al., 2016), some buy some commercially sold solutions containing a pheromone, with the disadvantage that it is more expensive and has a light gray color.

To identify the pests, the grower usually took the insects to an expert agronomist but that increases both production costs and response time until crop spraying begun.

After the advent and widespread use of digital cameras, a system using rural telecommunication infrastructure and computers was proposed. Farmers take digital photographs of plants and pests and send them using a computer and dial-up modem to experts that could give them an almost real-time response. (Koumpouros et al., 2004)

The biology, ecology and control of *Bactrocera oleae* has been thoroughly studied by a number of researchers during the last decades. Among them, those who refer to *B. oleae* control are of specific interest for this study and mainly include: cover sprays applied from the ground and the air, sterile insect techniques, mass trapping, biological control and bait-sprays applied from the ground.

Each one of them has its own advantages and disadvantages. The group of sterile insect techniques, mass trapping and biological control methods, are indeed environmentally friendly but they have been criticized for their somewhat reduced effectiveness and/or sufficiency as stand-alone methods in the case of non isolated olive-groves, or even in cases of specific climatological conditions (Broumas, 1995) (Varikou, Alexandrakis, Mavrotas, & Kouletakis, 2004) (Varikou, Alexandrakis, & Kalaitzaki, 2004) (Varikou, Alexandrakis, Mavrotas, et al., 2004) (Varikou, N., & Birouraki, 2014). On the other hand, cover sprays applied from the ground are very effective against *B. oleae*. Nevertheless, they have been criticized for a number of serious issues such as chemical burden of the wider ecosystem, soil and/or groundwater. In addition, the systematic use of cover sprays (of any kind) have been related with problems such as the chemical burden of the wider agro-ecosystem of olive-groves, the kill of the beneficial entomofauna, downgrading of human's health and safety working conditions and resistance to insecticides of the fly, as well. The interested reader is referred to (Broumas, 1994) (R. M. Ruiz-Torres & Muñoz-Cobo, 1997) (M. Ruiz-Torres & Montiel Bueno, 2002)

(Hawkes, Janes, Hemingway, & Vontas, 2005) (S. A. P. Santos, Pereira, Torres, & Nogueira, 2007) (V. M. R. dos Santos, Donnici, DaCosta, & Caixeiro, 2007) (Skouras et al., 2007) (Kakani & Mathiopoulos, 2008) (Doitsidis et al., 2016) and references there in, for detailed studies on cover sprays, their effectiveness and side-effects. Chemical residues in the produced table olives (Gilbert-López, García-Reyes, Fernández-Alba, & Molina-Díaz, 2010) and olive-oil (Amvrazi & Albanis, 2009) (Angioni, Porcu, & Pirisi, 2011) as well, have also been reported in the literature after intensive and no proper use of registered insecticides.

The protein-based bait-spray method was mainly developed due to issues related with serious side-effects on the environment and on human's health caused by cover sprays. The bait system, consists of an appropriate framework for Integrated Pest Management (IMP) as it manages to reduce pesticide levels with simultaneous beneficial results for predators, parasitoids and pollinators (Varikou, Garantonakis, Birouraki, Ioannou, & Kapogia, 2016). The method complies with the 2009/128/CE and 91/414/EEC directives regarding mass trapping and attract-and-kill methods for fruit-fly control (Varikou et al., 2016). It also complies with the EPPO/2012 standards ("Bactrocera oleae - bait application," 2012), with the National legislation and with the EU Common Agricultural Policy (CAP) regarding the challenges of the agricultural sector among the whole food chain length, the achievement of viable, economical and sustainable rural development. With this method, only a small part of the tree's canopy is sprayed with the insecticide solution (approximately 300 *cc*/tree, one tree after another or even one row after another of the orchard).

Table-olives and olive-oil's high prices motivated the farmers to intensify their production, thus maximizing their financial benefits and the environmental side-effects as well, as previously mentioned. Yet, the recent revision of the EU's rural and/or environmental policies will massively affect the viability of farming. It will promote biological and integrated control farming systems – such as the protein-based bait system of the olive-fruit fly control – which comply with sustainable farming in order to tackle with issues related to environmental burden (soil, olive products, water resources), and thus lead to safe growing practices and to the production of high quality products, which are indisputably preferred by the consumers.

Olive-tree cultivations have been extended to countries outside the EU, like U.S.A., Australia, Egypt and China during the last decades. They are characterized by a large production potential and by very well organized standardization and promotion networks, thus intensifying competition in the global market. Consumers (world-wide) gradually tend to consume food that is safer and of high quality. Thus, globalization and the challenges it brings, will inevitably affect olive-tree cultivation in a world scale.

Greece is dominated by an intense sense for effective environmental protection. Thus, *B. oleae* management is centrally managed and funded by the Ministry of Rural Development and Food and Regionally/locally supervised by the Directorates of Agriculture and Veterinary, and includes both the monitoring of the insect's populations and the application of bait-sprays (only) from the ground. Its annual cost approaches twenty (20) million euros (Pavlidi et al., 2013); (Varikou et al., 2016). Generally, three (3) bait sprays are applied during summer time and two (2) sprays during autumn depending on the olive-fruit fly population and infestation. Water soluble insecticides are exclusively used after September in order to minimize the existence of residues

in the produced olive-oil (Varikou et al., 2014). To monitor the olive-fruit fly population, standard glass and/or plastic McPhail traps are currently utilized (Figure 1 on the right). They contain a 2% of ammonium or protein solution in order to attract the flies. They are placed within the tree's canopy and the captured insects are counted every 5 or 7 days. When the mean number of captured adults per trap and per week is more than 8-10 combined with a few other criteria (sex ratio, climatological conditions, etc.), bait-sprays are then applied with the registered insecticide. Sprays are applied mainly by local owners of small-scale tractor and/or back sprayers. The spraying solution consists of the registered insecticide mixed with 2% of hydrolyzed protein (bait). A detailed evaluation of the attractiveness of various bait spraying solutions and their results may be found in (Varikou, Garantonakis, & Birouraki, 2015).

2.2 Automated Image acquisition and insect counting

Humans have predominantly been checking bait traps counting and identifying insects, but counting insects manually is a repetitive, time-consuming task, that leads to fatigue and can introduce errors (Yao et al., 2012), researchers have been trying to make systems detect count and sometimes even identify the species of the insects automatically. (Boissard, Martin, & Moisan, 2008), (Bechar, Moisan, Pulsar, & Antipolis-mediterranee, 2010), (Yao et al., 2012), (Potamitis et al., 2014), (Doitsidis et al., 2016).

In the dawn of the era of precision agriculture, it is imperative to develop systems that can automatically count pests, which is not a novel concept, it has already been researched in rice fields (Yao et al., 2012) and in greenhouses, (Solis-Sánchez et al., 2011) in somewhat controlled conditions.

The same problem has been tackled and groundbreaking work has already been made by a team of researchers in a joint effort to efficiently solve the difficult problem of determining the insect population and the best time to, for example, begin pesticide spraying. (Doitsidis et al., 2017), and that is what this thesis is trying to build upon.

3. MATERIALS AND METHODS

The system was built around a Raspberry Pi3 that functioned as the central processing unit of the trap. It is fitted with several sensors, that were connected and programmed one by one as they were made available.

The main sensor used for the purposes of this thesis is the camera, but a lot of other environmental sensors were added to the system to aid in the collection of data that could help reveal interconnections with insect population and environmental factors. All that can only be done if we can collect data for long time periods so that they can be of statistical importance.

In addition, some environmental sensors can aid in the decision-making aspects of the pesticide spraying, for example, to avoid strong winds when spraying.

The first tests were conducted offseason, using a small Olive tree plant, and while the pest population was not in large numbers. For the McPhail trap liquid solution, our initial test we did using an olive fruit fly pheromone solution bought from an agricultural supplies store, but when this liquid was viewed with the camera it made it more difficult to identify insects especially in low lighting conditions.

a) 

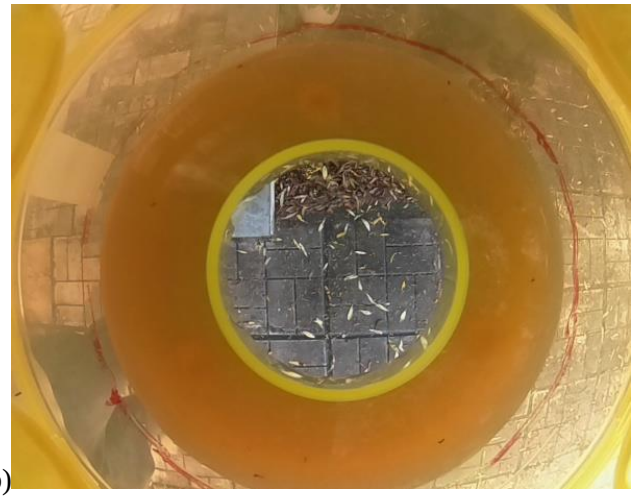
b) 

Figure 2. a) Pheromone solution, Bactrocera oleae (Gmelin) Dacus (Diptera: Tephritidae) attractant, b) image acquired with machine vision using the pheromone solution (light brown color)

Although a lot of effort was put to make this work, after many iterations of trial and error, a decision was made to revert to using a transparent liquid solution that does not interfere with the machine vision operations (2% water solution of ammonium sulfate, as mentioned earlier) this is discussed in detail in Computer Vision, bad design ideas.

During the development period, our trap was functioning using a 12Volt car battery until it was depleted and then recharged and the cycle run again and again. That is because the Raspberry Pi3 does not currently have Advanced Configuration and Power Interface (ACPI) support; meaning it cannot enter low power mode to conserve battery power.

3.1 Hardware and software overview

Although the raspberry pi has no onboard sensors, it provides a multitude of ways to connect them. Having first tried to connect a DHT11/DHT22 Temperature sensors to the digital GPIO pins, a realization was made, that reading the value from the data pin must be done at specific time intervals, and since that is accomplished by using delay timers (programmatically), achieving the precise timing is a somewhat problematic procedure and initial trials revealed that 25 percent of the time the values could not be read.

To both avoid sensor reading errors and more importantly avoid using extra GPIO pins for every sensor to be connected to the system, a choice was made to use sensors that support the two-wire Inter-Integrated Circuit bus (I2C).

During the initial testing phase as seen in the following Figure 3. Initial tests of the system component, sensors connected using a breadboard. using jumper wires and connected to the Raspberry Pi using a T-Cobbler Plus GPIO Breakout board.

This first setup was created using a regular 12 Volt car battery, a USB charger connected to the 12-volt accessory outlet (‘car cigarette lighter’) providing us the 5VDC to power the Raspberry, a breadboard and jumper wires.

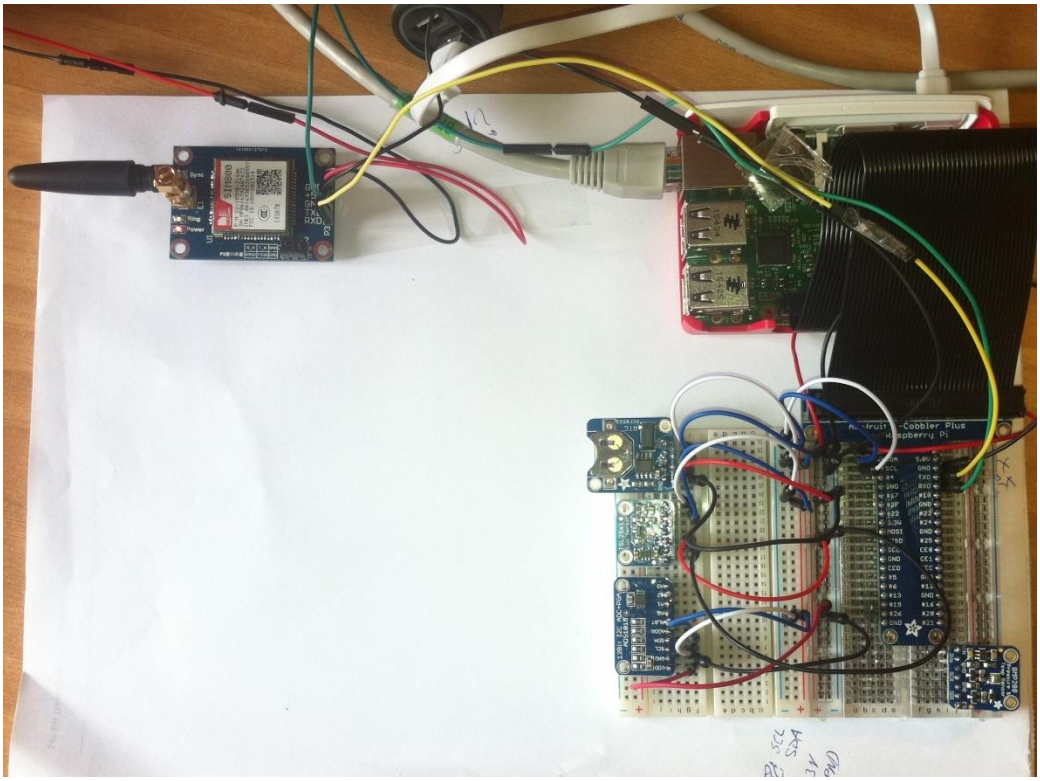


Figure 3. Initial tests of the system component, sensors connected using a breadboard.

In the following paragraphs, we are going to describe the various components in detail.

3.1.1 Raspberry Pi 3 single board computer

The Raspberry Pi is a low cost, single board computer, its processing unit is a Broadcom BCM2837 System on a Chip (SoC) that includes quad-core (4×) ARM Cortex-A53, 1.2GHz processor and an on-chip graphics processing unit (GPU, VideoCore IV). The board is equipped with 1 Gigabyte of RAM, one MicroSDHC slot and 40 General-purpose input-output (GPIO) connectors.

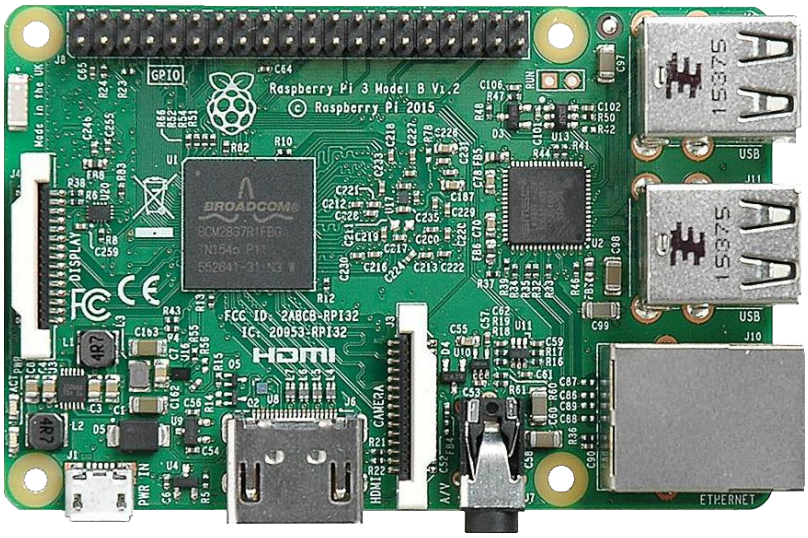


Figure 4. The Raspberry pi3, Single board computer

3.1.1.1 Operating system Setup

The Raspberry Pi's (Rpi) official supported operating system (OS) is Raspbian. At the time of this writing, the latest version, is Raspbian Jessi; which is a port of Debian Jessie for the Raspberry Pi ARM CPU that has out of the box support for all its hardware devices and buses**.**

The installation process is quite user-friendly, following the instructions on the official Raspberry Pi webpage (Raspberry_Pi_Foundation, n.d.), one needs to download the ZIP file with Raspbian Operating System Image "**`2017-04-10-raspbian-jessie.zip`**" and **ETCHER**, an application (available for Windows, Linux, and MacOS) that is used to transfer the Raspbian image onto the microSD card. A screenshot of the application transferring an image on a microSD card is seen bellow.

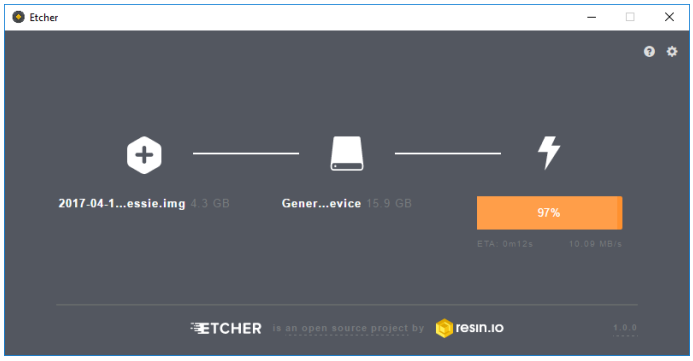


Figure 5. Writing the Raspbian Operating System on the microSD card

After inserting the microSD card in the Rpi the file system is expanded to the maximum microSD card capacity, the system reboots again, entering a Graphical User Interface. After the initial testing phase, the boot configuration can be altered, disabling the X-server to make the OS, boot to a command prompt without any GUI, reducing boot time.

The first thing that should be done is to update the operating system to incorporate any security patches, that is done by opening a terminal window and executing the commands as shown in the Appendix A.1. Updating the Operating System.

In order to set up the CSI interface (camera), SPI and I2C bus (sensors) one can use either the terminal, by using the command: **`sudo raspi-config`** or using the graphics environment by selecting from **`Applications Menu > Settings > Raspberry Pi Configuration.`**

The final raspberry pi configuration settings are shown in the following figure.

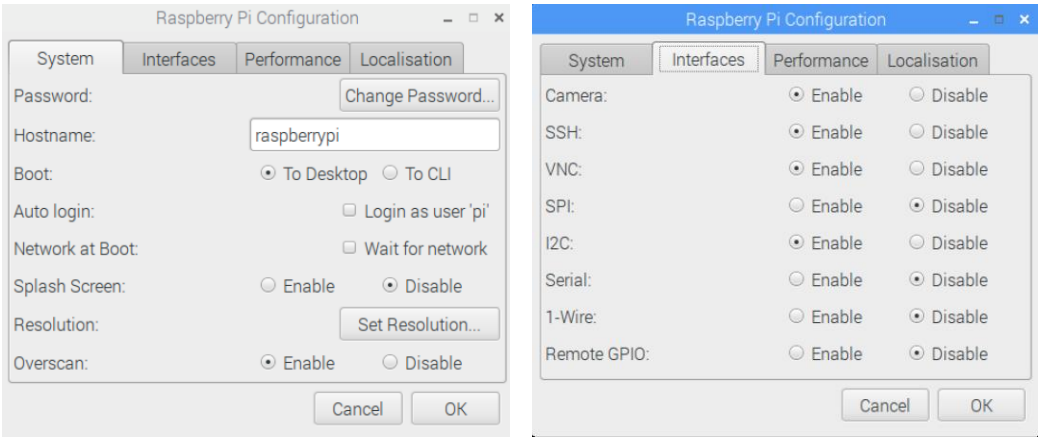


Figure 6. Raspbian system and interface configuration screen

Apart from the above initial configuration changes, other customizations are needed to enhance the security and allow for the proper operation of the trap.

Some of these customizations are:

- Change the default passwords for users **`pi`** and **`root`** and enable SSH key authentication for security reasons.
- Change the system time zone and make the appropriate changes to use the Real-Time Clock module, so that at boot we don't need to have an internet connection to know the current date and time.

- Enabling the camera and i2c bus modules at boot time, setting a fixed display resolution to be able to connect to a headless raspberry pi remotely, detailed instructions are shown in the Appendix: *A.2 Enabling kernel modules at boot time.*[1]
- Install most commonly used terminal utilities like **`vim`**, **`wget`** etc, as shown in the Appendix: A.3 Install some command line utilities and customizations.
- Add scheduled tasks that are executed at defined intervals. Such tasks in Linux are called **`cron jobs`** and we use them to
    - enable the system to *send status emails (described in Appendix:* A.4 Sending email report when unit has internet connectivity*),*
    - synchronize internet time with local hardware clock time (described in Appendix: A.6 Automatically update the HW-clock with NTP Internet server) and of course
    - automate the execution of the automated trap sensor data acquisition (described Appendix: A.7 Python scripts that collect sensor values and send them to the webserver*).*

[1] Note: After any changes are made in /boot/config.txt, the operating system needs to be restarted for them to take effect.

3.1.1.2. Data Storage

Raspberry Pi does not have built in storage, the operating system (OS) is copied on a MicroSD card out of which the Raspberry Pi boots. The boot image is transferred onto the MicroSD Card using another computer. The process of creating the bootable MicroSD card is covered in the section "Operating System Setup".

The same MicroSD card is used as a primary OS disk, storing essential operating system files, the swap file and of course user files. We also need to temporary store information for data logging purposes, on the local media as we read values from the various sensors. Keeping a local copy of the sensor data provides a backup copy in case of power or communication failure with the remote web server. But constantly changing OS log files or data logger files push the microSD card to its limits.

All SD cards have what is called maximum read/write cycles (number of times that the contents of the disk can be altered). The typical limit imposed by current technology and manufacturing processes and quality is between 10.000 - 100.000 read/write cycles.

Having all the above in mind, it is recommended, to replace the SD card used at regular intervals (about once a year), with a fresh one as a precaution. Thus, avoiding any malfunction or data loss from the affected system.

For the purposes of this thesis all files were created and kept in the root folder **/trap** and in there, some other folders exist like: **/trap/cpp, /trap/grabbed, /trap/buffer, /trap/buffer/processed** and others named in an intuitive way to reflect their contents.

3.1.2. Camera / Machine vision sensors

3.1.2.1 Camera selection

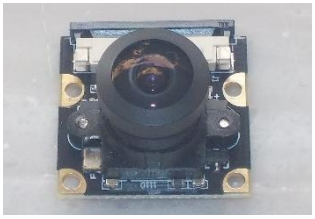
Another important sensor for this system is the vision sensor. We had several candidate cameras to test, to determine if they could be used as vision sensors. From the sensors tested, one camera taken from an LG Smart TV, a Sony PlayStation 3 Eye Cam, two Logitech USB web cameras and the raspberry pi CSI fisheye camera.

a) LG Video Call Camera (AN-VC500)

b) Sony PlayStation 3 Eye Camera

c) Logitech QuickCam Communicate STX

d) Logitech HD Webcam C270



e) Raspberry Pi CSI Fisheye Camera

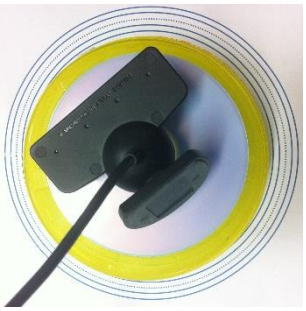
Figure 7. cameras tested a) LG Smart TV USB camera, b) Sony PS3 Eye camera (two settings for focal length), c) Logitech QuickCam Communicate STX, d) Logitech HD Webcam C270, e) Raspberry Pi Camera (wide angle lens)

Each camera was connected and a test image was taken, during this phase, we realized the: LG Smart TV USB camera did not function at all, probably due to lack of drivers for the Raspbian OS, all other cameras functioned properly.

One of the first problems was positioning the cameras on top of the trap lid and properly aligning them at the center. Apart from the Raspberry Pi CSI Camera that is small and has a flat surface, all other cameras due to their different outer dimensions and plastic covers could not easily rest flat at the top.

Images of the test conducted placing the cameras and taking sample images from each one are seen in the following Figure 8. Camera placement in the top lid of the McPhail trap a) Sony PS3 *Eye* camera (two settings for focal length), b) Logitech QuickCam Communicate STX, c) Logitech HD Webcam C270, d) Raspberry *Pi* Camera (wide angle lens)

All cameras were placed at a fixed distance from the McPhail trap top lid and an image was captured.



a) Sony PlayStation 3 Eye Camera placement top view

a2) side view

a3)images Captured

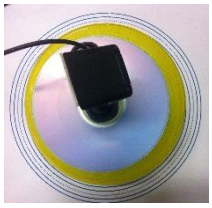
a) red dot (normal)

b) blue dot (wide angle)

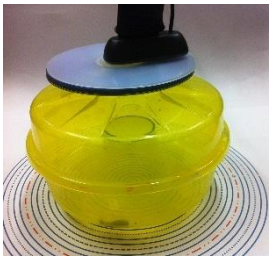
b) nicate STX

b2) side view

b3) image Captured

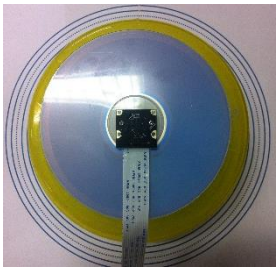


c) Logitech HD Webcam C270



c2) side view

c3) image Captured



d) Raspberry Pi CSI fisheye Camera

d2) side view

d3) image Captured

Figure 8. Camera placement in the top lid of the McPhail trap a) Sony PS3 Eye camera (two settings for focal length), b) Logitech QuickCam Communicate STX, c) Logitech HD Webcam C270, d) Raspberry Pi Camera (wide angle lens)

It became clear that both Logitech Cameras tested, were designed to be used for web conference meetings and their lenses were optimized to capture objects that are one or two meters away. The Sony PS3 Eye camera with the blue dot (wide angle) setting captured a bigger area that the Logitech web cameras but it still left some blind spots that were not in the frame.

As seen in Figure 8, *d3) image captured*, the Raspberry Pi Fisheye camera has the greatest field of view of all tested cameras and is easier to mount on the trap top lid.

The Raspberry Pi camera, during the initial/testing phase, was mounted on the top lid just protruding through the center hole of 8 CDs see details of Figure 8. This setup was optimum and was the one we used for our experiments hereon.

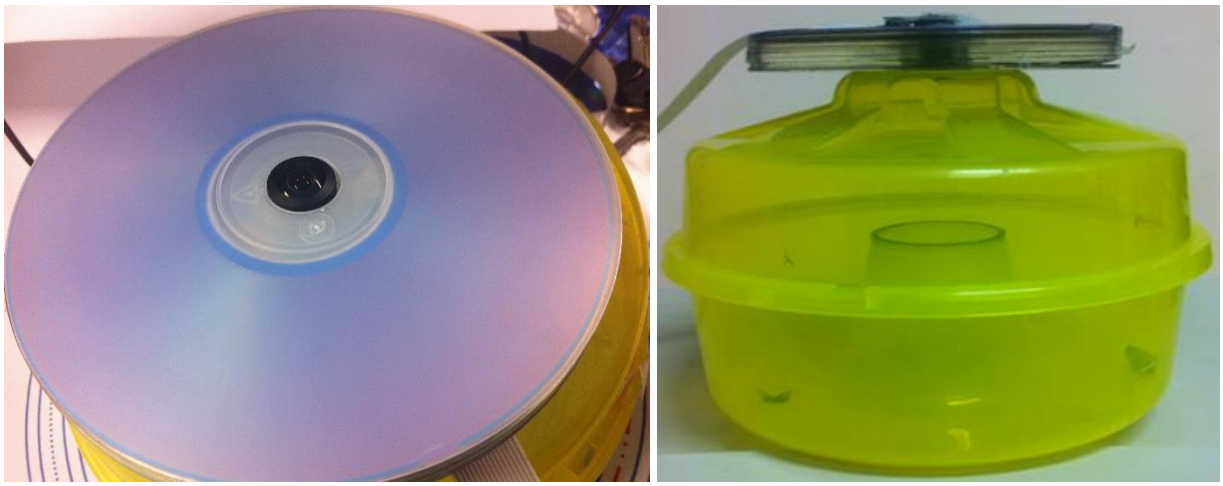
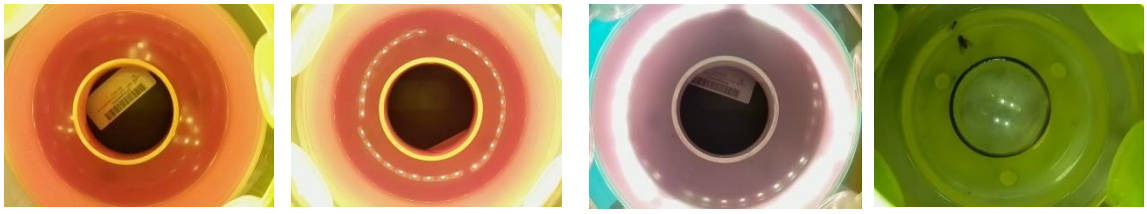


Figure 9. Camera placement Raspberry Pi Camera Fisheye (wide angle lens)

3.1.2.2 Artificial lighting

To be able to take photos of the trap surface at night time, led lights were placed on the top half of the trap, but immediately reflections were seen on the image that could lead in more difficult image processing later. To avoid reflections on the liquid surface many led positions were tried, as seen in the next figure:



a) *b)* *c)* *d)*

Figure 10. Different arrangements of LED Lights on the top half of the plastic lid of the trap a) triangle shape, b) circle shape, c) diffuser with a circle shape, d) next to the camera lens

It became clear that the best position was as close to the center as possible, around the protruding camera lens. Although it would be preferable to shed as much light as possible, there just isn't enough space to fit more than 10-15 LEDS on the top lid without having them reflect on the liquid surface.

The final position selected can be seen in Figure 11 a) and b) in detail. LED lights have been positioned in two areas, in a circular fashion, facing outwards around the top hole of the trap, and laid flat facing downwards around the camera lens. This was more difficult to achieved but provided significantly more lighting to the scene the camera was shooting.

a)  *b)* 

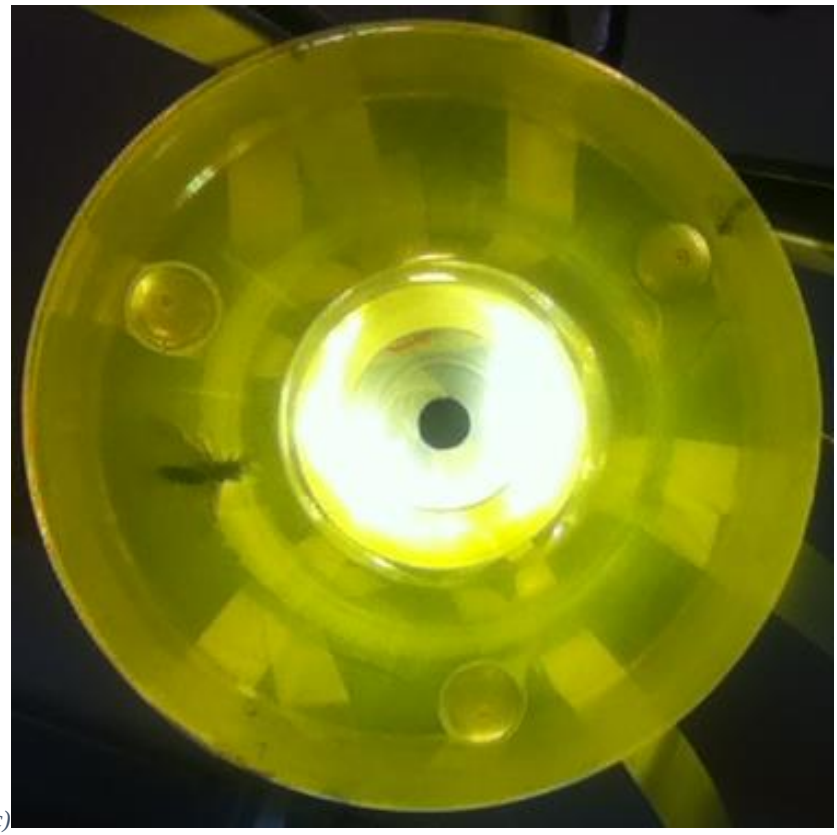
c) 

Figure 11. Final positioning of LED Lights, a) LED's powered off upper part of trap only, b) LED positioning on upper part detail, c) LED's powered on

3.1.3. Real Time Clock (Adafruit PCF8523 I2C)

This device is a battery-backed Real Time Clock (RTC), it connects to the Raspberry Pi with the two-wire I2C bus and allows the Raspberry Pi to keep time without having to first connect to the internet.

Raspberry and its official OS Raspbian uses a software clock and synchronizes from an Internet NTP server, when we shut down the OS a file is written to the OS disk that is read at boot time, the Raspberry Pi has no concept of absolute time and will populate the system time with the date and time read from that file. used without a hardware Real Time Clock and not knowing how long it has been turned off, would lead to an erroneous estimate of current date and time without first connecting to the internet.

Therefore, this is an essential part of the design that allows the system to know the date and time immediately after booting without having to contact a Network Time Server (NTP) first; that way we can read and store data from the sensors with correct timestamps at times when we do not want or cannot connect to the Internet.

On the other hand, if the system connects to the internet only at predefined intervals, we could also prolong the battery lifetime.

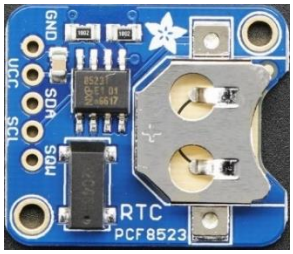


Figure 12. The Adafruit PCF8523 Real Time Clock Assembled Breakout Board, photo from *(Adafruit RTC, 2016)*

In order to utilize this module, the default behavior of the Raspbian OS described above, has to be overridden.

Detailed instructions about the procedure that must be followed are at the Appendix: A.5 Enable the I2C Real Time Clock module. One thing to note is that if during the operating system boot time the RTC cannot be discovered on the I2C bus, then it is not used as a system hardware clock, until the next time the system is restarted.

3.1.4. BMP280 Barometric Pressure and Temperature Sensor Breakout

This breakout board incorporates the Bosch BMP280 sensor, it connects to the Raspberry Pi with the two-wire I2C bus and gives readings for Barometric Pressure and Temperature.

Figure 13. The Adafruit BMP280 Barometric Pressure + Temperature Sensor Breakout photo from *(Adafruit BMP280, 2015)*

The BMP280 sensor measures Barometric Pressure with ±1 hPa absolute accuracy, and Temperature with ±1.0°C accuracy. Indirectly this sensor, using the pressure reading, could also calculate the altitude ±1 meter or better accuracy, but it would need a pressure reading at sea level (at that specific time) as a reference.

This sensor board SCK pin 4 is connected to the clock SCL of the I2C bus and the SDI pin 6 to the I2C data SDA.

After first connecting the sensor using the I2C bus, and using a test program we get the first readings: ***Pressure: 988.96 hPa, Temperature: 21.72 C (Celsius), Temperature: 71.10 F (Fahrenheit)***

3.1.5. The AOSONG encased I2C Temperature/Humidity Sensor (AM2315)

This sensor connects to the Raspberry Pi with the two-wire I2C bus and two 10K pull-up resistors and gives readings for Temperature and Humidity. The sensor update rate is 0.5Hz (once every 2 seconds) and Humidity is measured with 2% accuracy and Temperature with ±1.0°C accuracy.

Although it looks, and is more, rugged than the BMP280 breakout board, it should not come in direct contact with water, rain or sunlight because it will either give wrong readings or be permanently damaged. This sensor does not come ready made with a breakout board. Two 10K pull-up resistors were used to connect it to the I2C bus. The red wire is connected to 3.3V power, black to ground, yellow wire to i2c data pin (SDA), and the white wire to i2c clock pin (SCL) as seen in the following figure (Sopwith, 2014).

Figure 14. The AM2315 - Encased I2C Temperature/Humidity Sensor, photo from *(AdafruitAM2315, 2013)*

As shown in Figure 15. Connecting the AM2315 Humidity/Temperature sensor to I2C bus (red 3.3V, black GND, yellow i2c SDA, white i2c SCL), two ~10Kohm pullup resistors are required to connect it to the I2c Bus (Between the SDA and SCL lines and the- power wire), and it appears on the I2C bus with the default 7-bit I2C address of **`0x05c`**.

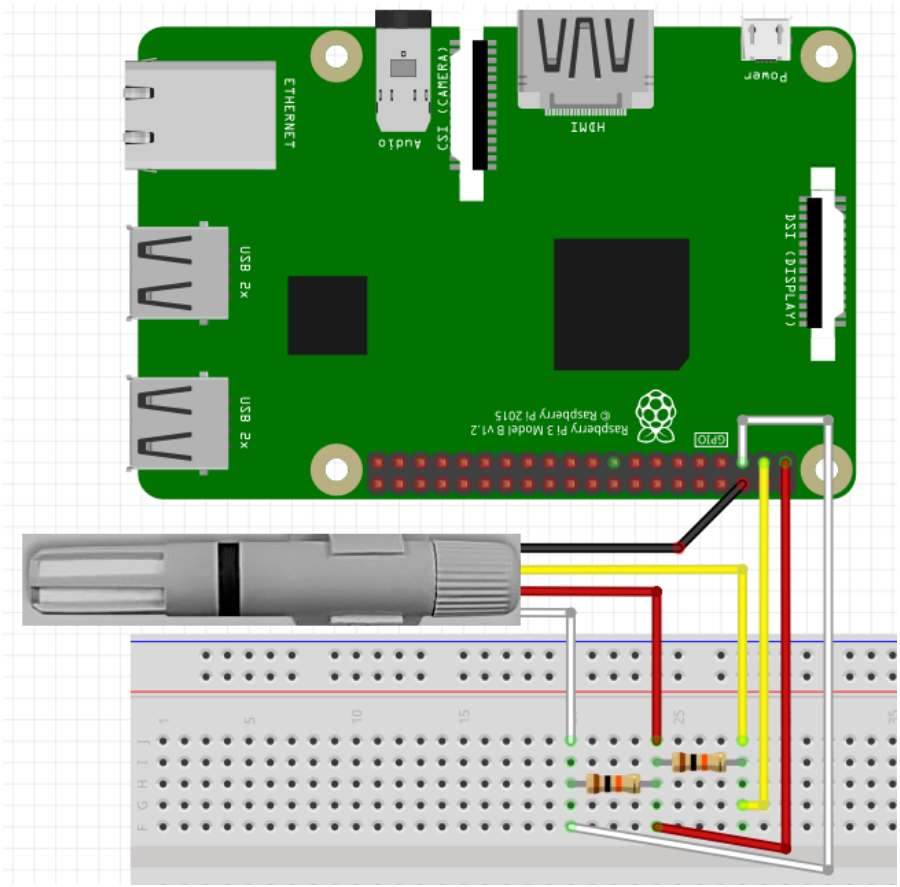


Figure 15. Connecting the AM2315 Humidity/Temperature sensor to I2C bus
(red 3.3V, black GND, yellow i2c SDA, white i2c SCL)

3.1.6. Anemometer Wind Speed Sensor w/Analog Voltage Output

This anemometer needs to be powered, with 7-24 Volts and its output in voltage levels is linear to the wind speed it is measuring. As it is stated in the manual, output of 0.4 Volts up to 2 Volts are for speeds between 0.5 m/s and 50m/s respectfully.

Figure 16. An image of the Anemometer Wind Speed Sensor w/Analog Voltage Output photo from *(Adafruit Anemometer, 2014)*

As stated in the ("Adafruit Wind Speed Sensor w Analog Voltage output (Product Manual)," 2016), the sensor can measure speeds up to 32.4 m/s which is about **12** in the Beaufort scale, that corresponds to hurricane force wind speeds (wikipedia.org, 2017), a speed that would never be expected to record.

The sensor needs to be supplied with: 7V ~ 24 V DC to operate, Pin 1 Power (**brown** wire), Pin 2 Ground (**black** wire), we are using the main 12V battery. And the signal outputted from this specific sensor is voltage in the range of 0.4 ~ 2V, measured from Pin 2 Ground and Pin 3 Signal (**blue** wire).

The mathematic equation that expresses the wind speed from the volts measured is:

Wind speed = ((output voltage -0.4) /1.6)*32.4

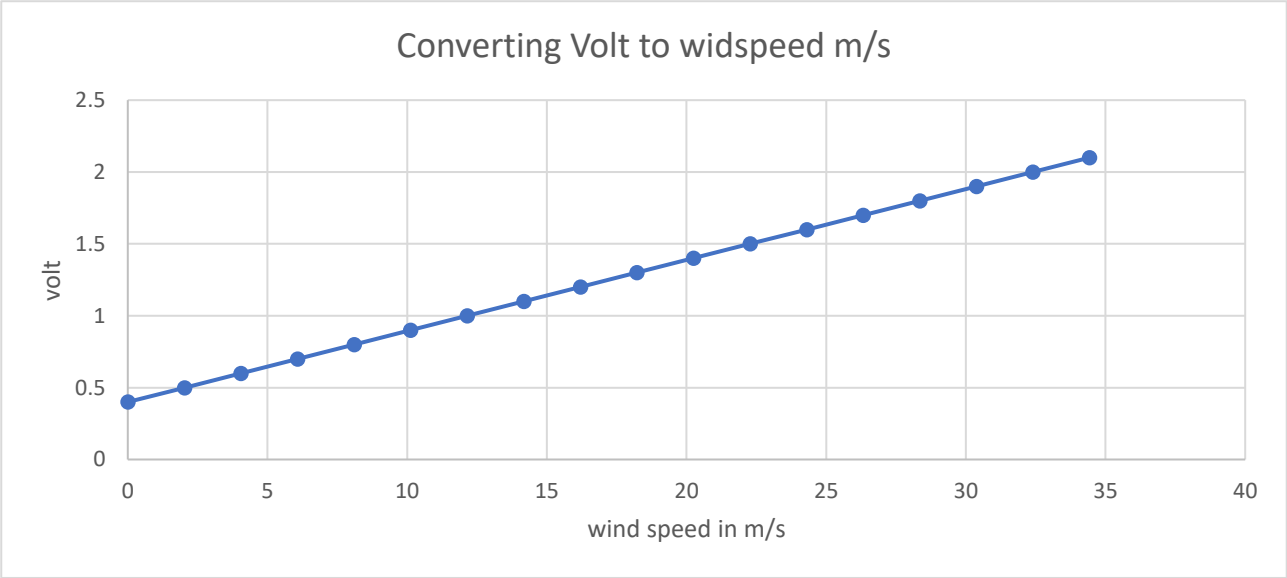


Figure 17. Converting volt output from the anemometer to wind speed expressed in m/s

To read the voltage levels we are using the Adafruit ADS1015 12-Bit ADC, described in the following paragraph, that handles the conversion of the analog voltage signals to readable digital value.

3.1.7. ADS1015 12-Bit ADC - 4 Channel with Programmable Gain Amplifier

As is was already stated, the Raspberry Pi does not have an embedded Analog to Digital Converter (ADC) circuit, therefore we need an external ADC to convert and read voltage values from analog sensors.

This specific ADC has 12-bit resolution and programmable sample rate of 128 to 3300 samples/second. Since we are using single ended inputs to measure the voltage between the analog input channel (A0-A3) and analog ground (GND), and we only measure positive voltages between ground potential and VCC and without the sign bit we effective only have 11 bit resolution (Earl, 2016). This breakout board can appear with 4 different addresses on the I2C bus, but since we are only going to connect one board to the bus, we are using the default chip 7-bit I2C address of **`0x48`**.

Although this ADC has four channels, we are going to use just two:

(a) one for the anemometer (to convert voltage values to human readable wind speed values, meters/second) (b) and another to read battery voltage levels to warn service personnel that battery is depleted and needs replacing.

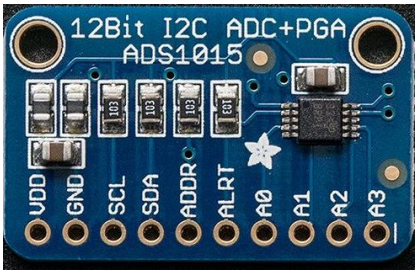
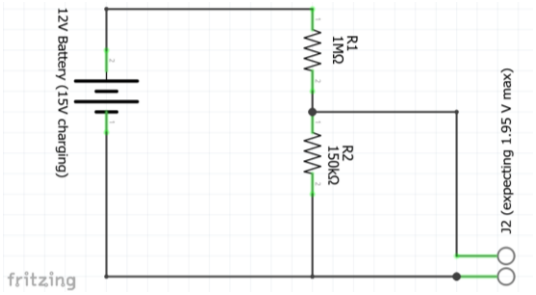


Figure 18. The ADS1015 12-Bit ADC - 4 Channel with Programmable Gain Amplifier photo from *(Adafruit ADS1015, 2012)*

a) To read the wind speed we convert voltage values read from the ADC using the mathematical expression already elaborated in previous paragraphs. To read the voltage from the anemometer (max 2V) we must use gain=2 on the ADS1015 ADC module for maximum resolution.

b) To read the battery voltage (max 15V while charging), we must use a Resistive Voltage Divider (RVD), to reduce it down to a max value of 1.95V, which is comparable to max the value read from the anemometer. To accomplish that we use the schematic as seen in Figure 19. A Resistive Voltage Divider created to reduce voltage to be measured from 15V to 1.95V.



R1: ~1003 KΩ

R2: ~149,4 KΩ

The two resistors due to tolerances (±5%) have created a slightly different voltage divider, measuring with a voltmeter we measured that when the car battery input voltage is 13.20 the output from the voltage divider is 1,516.

Therefore, in order to get the actual battery voltage, we should divide the measured value from the ADC with 0,114848 ~=**0.115**

Figure 19. A Resistive Voltage Divider created to reduce voltage to be measured from 15V to 1.95V

To read the ADS1015 we used the C++ driver for Linux found in the (Reignier, 2017) GitHub repository: Driver for TI's ADS1015: 12-bit Differential or Single-Ended ADC with PGA and Comparator.

3.1.8. I2C Digital Luminosity/Lux/Light Sensor (Adafruit TSL2561)

This sensor, when placed in a proper position with unobstructed view of the sun's rays, it can measure a lot of useful light parameters, that can be used to indicate if it is day or night thus help us determine if we need to use to turn on led lighting when taking pictures with the camera sensor.

Figure 20. The Adafruit TSL2561 Digital Luminosity/Lux/Light Sensor Breakout photo from *(Adafruit TSL2561, 2014)*

After first connecting the sensor using the I2C bus, and using a test program we get the first readings:
Test. RC: 0 (Success), broadband: 2515, ir: 1022, lux: 127

3.1.9. SainSmart SIM800 GPRS/GSM Board Quad-Band Module Kit

This is a GPRS/GSM module that interfaces with the Raspberry Pi using the UART serial interface, but several problems were encountered when trying to connect this module to the Raspberry Pi.

The power input of this module is 5V with Peak current of 2A, that means that we need to have a dedicated power supply for it. After reading the specifications for the SIM800 chip it stated that it can operate from 3.4 to 5V, so the power supply used, a variable voltage regulator was set to provide 3.45V.

The initial decision to power the board with the minimum operating voltage was made to avoid damaging the Raspberry when communicating using the TX/RX pins since is must have the same Ground potential reference and the same high and low values for the serial signal.

After connecting the device to an oscilloscope, we discovered that although the board seemed to be powering on, the RX/TX pins showed no activity. That led to the conclusion that although the SIM800 chip can operate with a minimum of 3.4 Volts the SainSmart board with all the electronics needed the full 5 volts to properly function.

Figure 21. The GSM/GPRS Module

After the input power adjustment, the board seemed to power on and showed activity in the TX/RX pins but we still could not get it to communicate with the Raspberry Pi.

Reading through various forums, many users reported similar problems while trying to communicate using UART1 pins 08 - GPIO 14 (TXD0) and 10 - GPIO15 (RXD0) as seen in Figure 54. Raspberry Pi GPIO Header Pinout. The source of the problem was that the Raspberry Pi3 has an onboard Bluetooth module that communicates through Serial0 (UART0) and that the Serial1 is a software UART, and its baud rate greatly relies on clock speed.

Some propose to limit CPU clock speed and use the software UART, while others prefer to not limit clock speed / computing power of the Pi and disable the Bluetooth module, if they do not have any use for it.

(Hughes, 2017)

```
root@trap:~# ls -lar /dev/seri*
lrwxrwxrwx 1 root root 7 Jan  1  1970 /dev/serial1 -> ttyAMA0
lrwxrwxrwx 1 root root 5 Jan  1  1970 /dev/serial0 -> ttyS0
```

The solution that was tested and communications between the raspberry pi and the SIM800 board was established was by changing the options in configuration file `/boot/config.txt` and adding the line `enable_uart=1`, using the /dev/serial0, software UART.

To test the connection, we used the minicom application and output from the session is shown in the following figure. The serial connection was made at 115200 baud,

`minicom -b 115200 -o -D /dev/serial0`

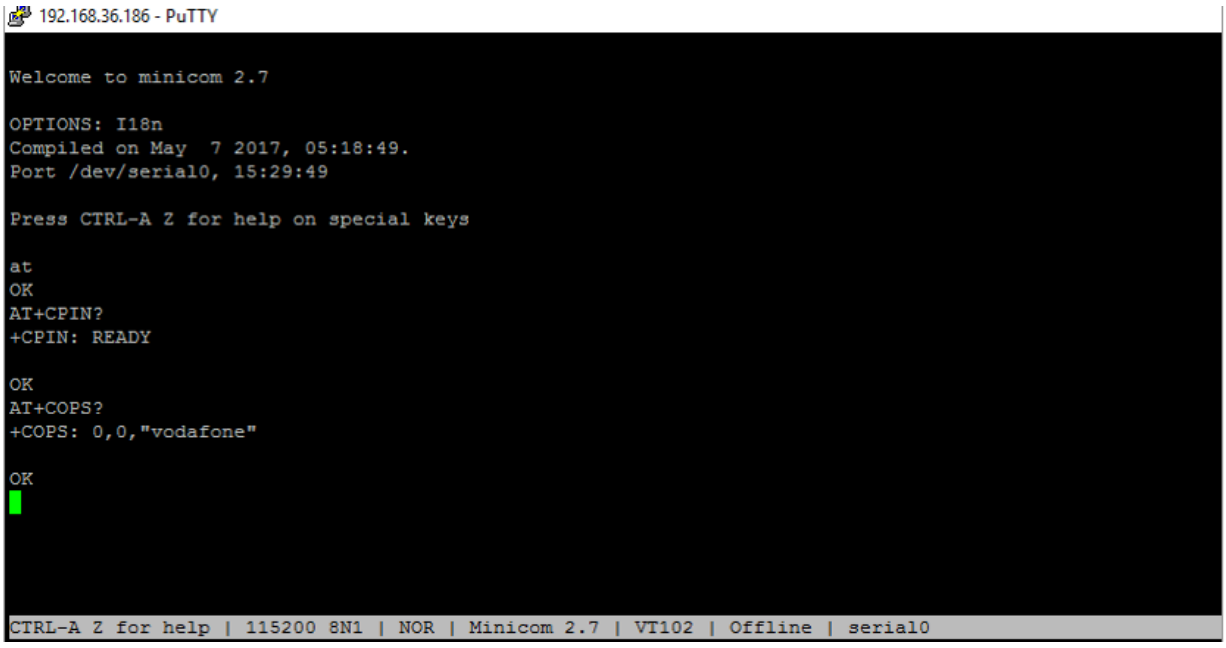


Figure 22. testing the Raspberry Pi serial communication with the SainSmart SIM800 GSM/GPRS module

Efforts to automate the connectivity to the internet and usage of the SainSmart SIM800 GSM module were made by following instructions for other Linux boards Like BeagleBoneBlack and Raspberry Pi2 though using different GSM modules ((DiCola, 2014))

The objective is to have the Linux operating system initiate a PPP connection and use the connection for regular internet TCP/UDP traffic and avoid using AT commands that have many limitations can only be used by our script. Using this method, remote Raspberry Pi stations could even perform operating system upgrades. As seen in the Appendix: Error connecting to the Vodafone GSM/GPRS Network.

3.1.10. Relay switch to turn on the LED lights, power of Anemometer and turn on the GSM modem

After verifying that in principle, the design is usable and works properly we decided to get rid of the breadboard and soldered the first custom board shown in Figure 23. It was used to power on and off the LED lights, the Anemometer and facilitated monitoring of the battery voltage (using an RVD) as well.

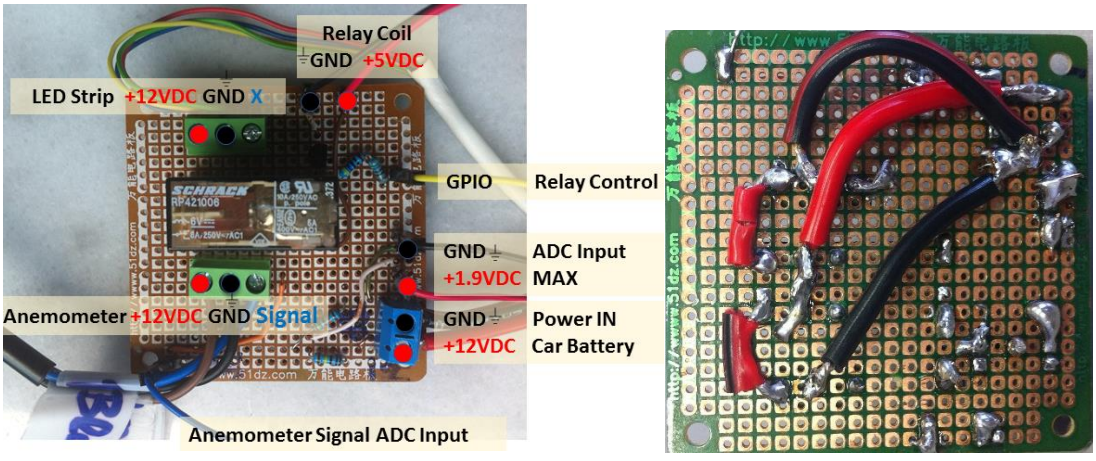


Figure 23. 1st Custom soldered board with marked inputs and outputs

3.1.11 I2C Bus

I2C is a useful bus that allows data exchange between microcontrollers and peripherals with a minimum of wiring. The bus itself has two wires, the clock line, and the data line, using just those two wires, we can connect, up to 128 devices when using 7 bit addressing or up to 1024 devices when using 10 bit addressing.

The bus usually has one master that provides the clock (SCL) and many slave devices, although on there are setups when a multi-master i2c bus is used. Using I2c Bus results in the processor having more GPIO pins free to be used with other accessories, sensors and actuators one might want to connect in the future.

Two things must be considered about I2C bus, a) device logic level and b) device addressing to avoid collisions. a) All signals should be converted to the signal level (voltage) used by the I2C Master. So all devices we connect must have the same logic level (3.3V). If we want to mix devices of different voltages, we must use an electronic circuit called "logic level shifter" that translates voltage levels to and from 3.3V and 5V allowing the connection of those devices on the same bus, such a configuration is shown in Figure 24. Connecting devices with different I2C logic level

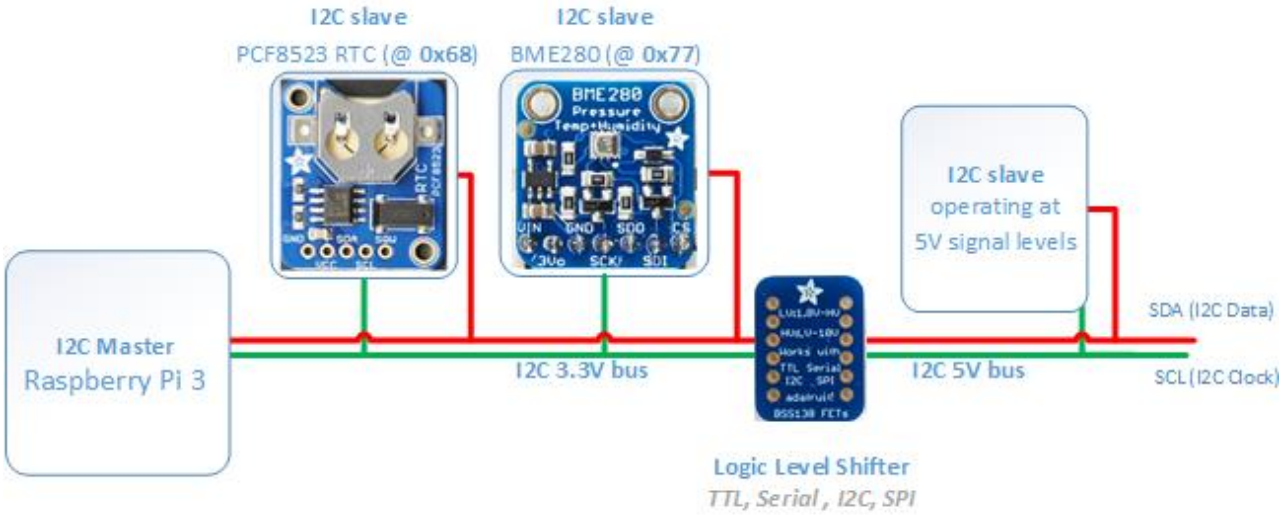


Figure 24. Connecting devices with different I2C logic level

b) Typically, I2C devices have a fixed address that cannot be changed by the user, therefore we cannot use e.g. two temperature sensors from the same manufacturer on the same i2c bus. There are exceptions where devices have jumpers that may allow for address selection example.

The Rpi 3 has two I2C Buses, but only one is fully functional (I2C 1), the other one is only used to connect other shields, HATs (I2C 0). The I2C bus connections of our Dacus Trap implementation can be seen in the Figure 25. The Raspberry pi3, I2C bus connections, where Raspberry PI 3 (I2C master) and sensors (I2C slave devices) are shown.

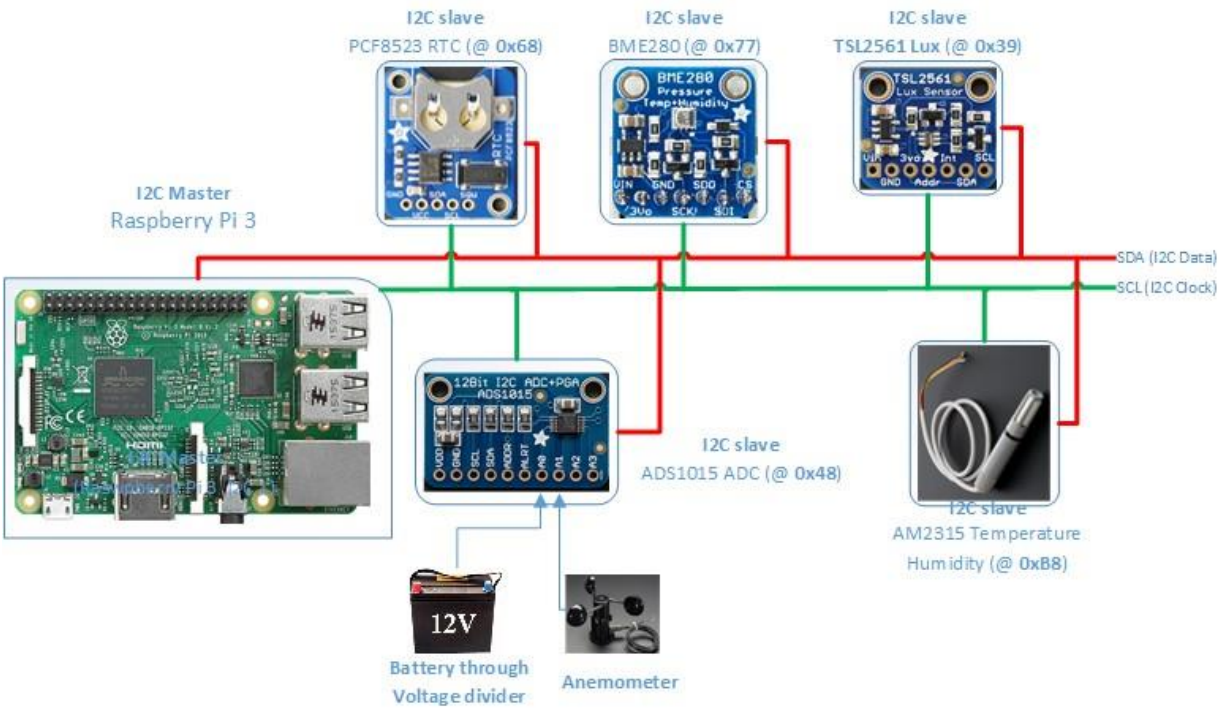


Figure 25. The Raspberry pi3, I2C bus connections

To show each device's address on the bus we can issue the command: **`i2cdetect -y 1`**

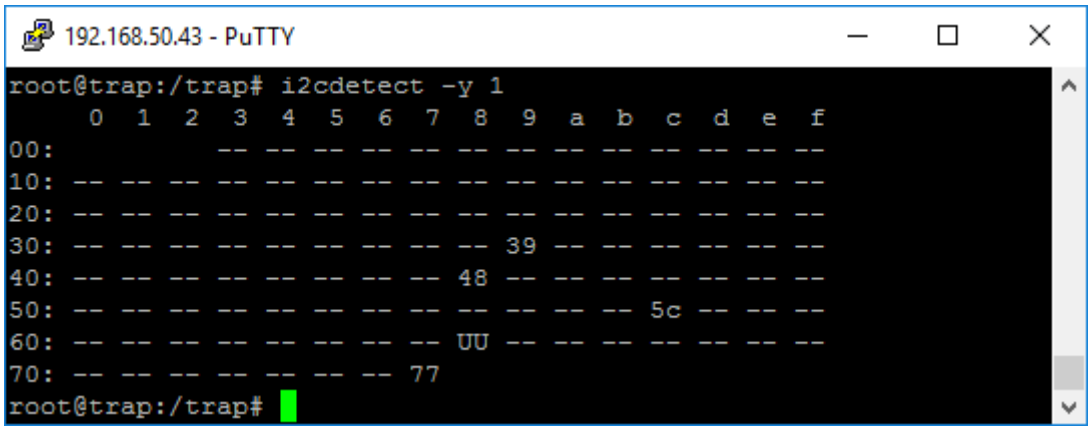


Figure 26. The Raspberry pi3, I2C bus command to detect devices on bus. "i2cdetect"

The devices detected on the I2C bus are:

Index	address	Sensor Description
1	39	TSL2561, Light Sensor (Lux, IR, Broadband)
2	48	ADS1015 ADC, Analog to Digital Converter
3	68	PCF8523 RTC, Real Time Clock (the address is shown as UU)
4	77	BME280, Pressure and Temperature Sensor
5	5c	AM2315, AOSONG encased Temperature and Humidity Sensor

In the paragraphs that follow, a closer look is taken at the various sensors used, their connection to the main system and basic programs used to query their values.

3.1.12 System as a whole

After verifying that in principle, the design is usable and our first attempt was successful, we decided to stop using the breadboard and design a new circuit board that would accommodates all sensors and components in a semi-permanent and modular way.

3.1.12.1 Block diagram

In the Figure 27, the system block diagram is represented. The system is divided into two parts. On the left-hand side, we can see a water proof enclosure (IP67 certified), that houses the main system components (raspberry Pi, battery, GSM Modem, the Voltage Regulators, and some sensors) and on the right-hand side we can see the remaining sensors that are on the top lid of the trap (LED lights, Light Sensor, CSI Camera), or at an even greater distance, placed on a pole (wind speed sensor).

This version of the trap, has some inherent design limitations due to the length of the CSI ribbon cable, that is two-meter long. All other cables can easily be made even longer because the I2C Bus is used to connect components inside buildings as well.

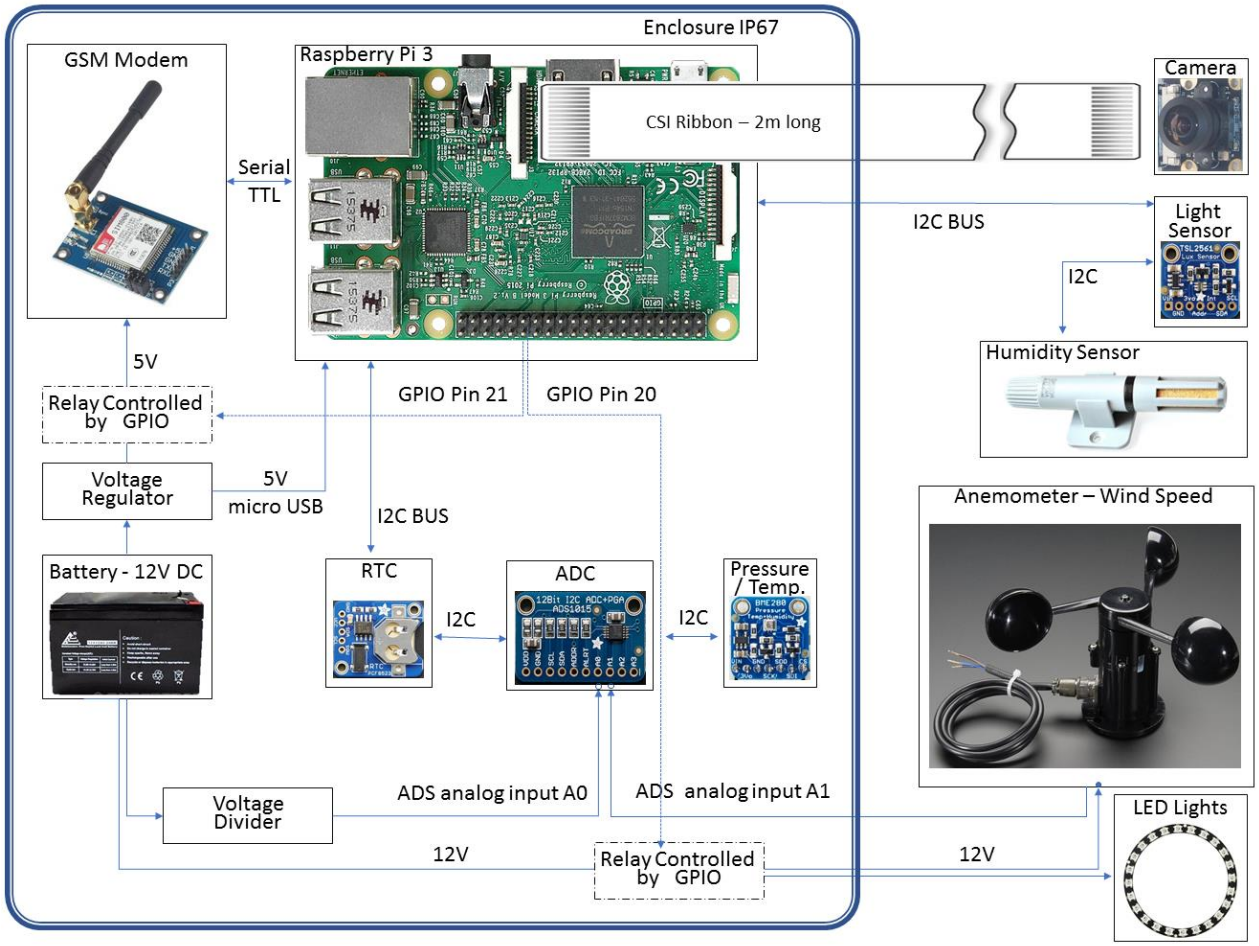


Figure 27. Complete system block diagram

3.1.12.2 Designing the circuit board to accommodate all sensors and components

Using the open source application “Fritzing”, http://fritzing.org, all components were placed and connected using a virtual breadboard.

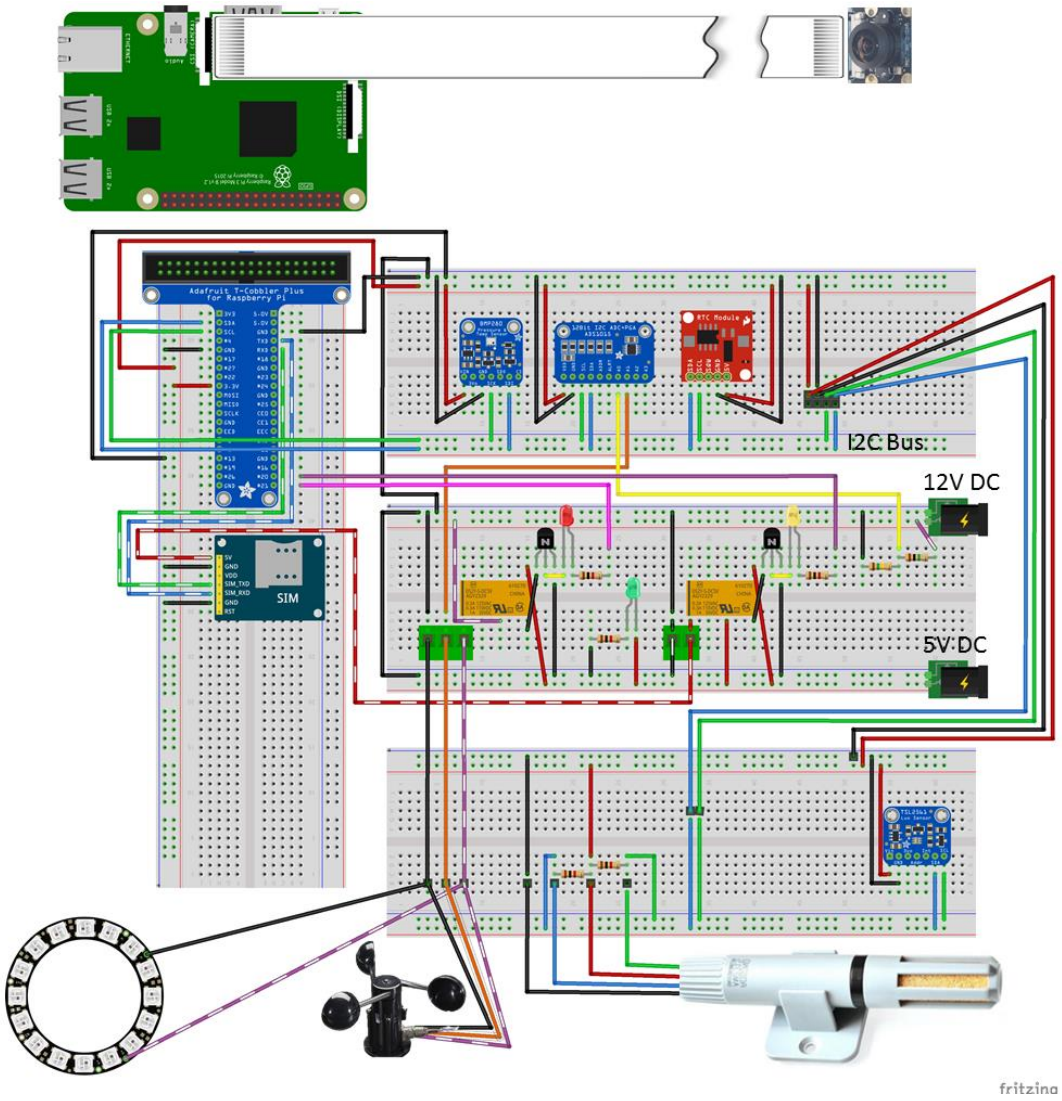


Figure 28. All components connected using the fritzing application

3.1.12.3 PCB Design

The same open source application fritzing was used to Schematics and PCB design has been made using the

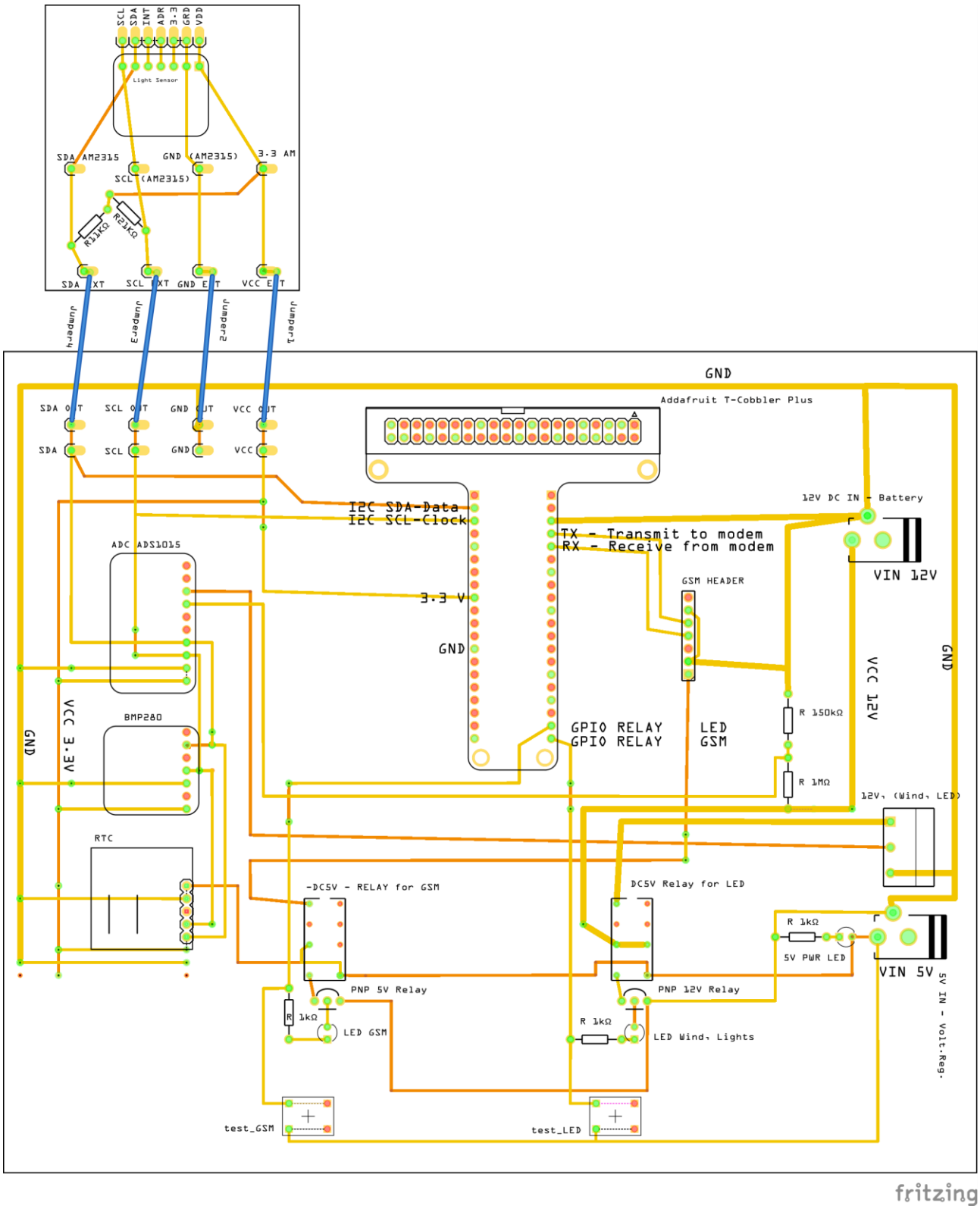
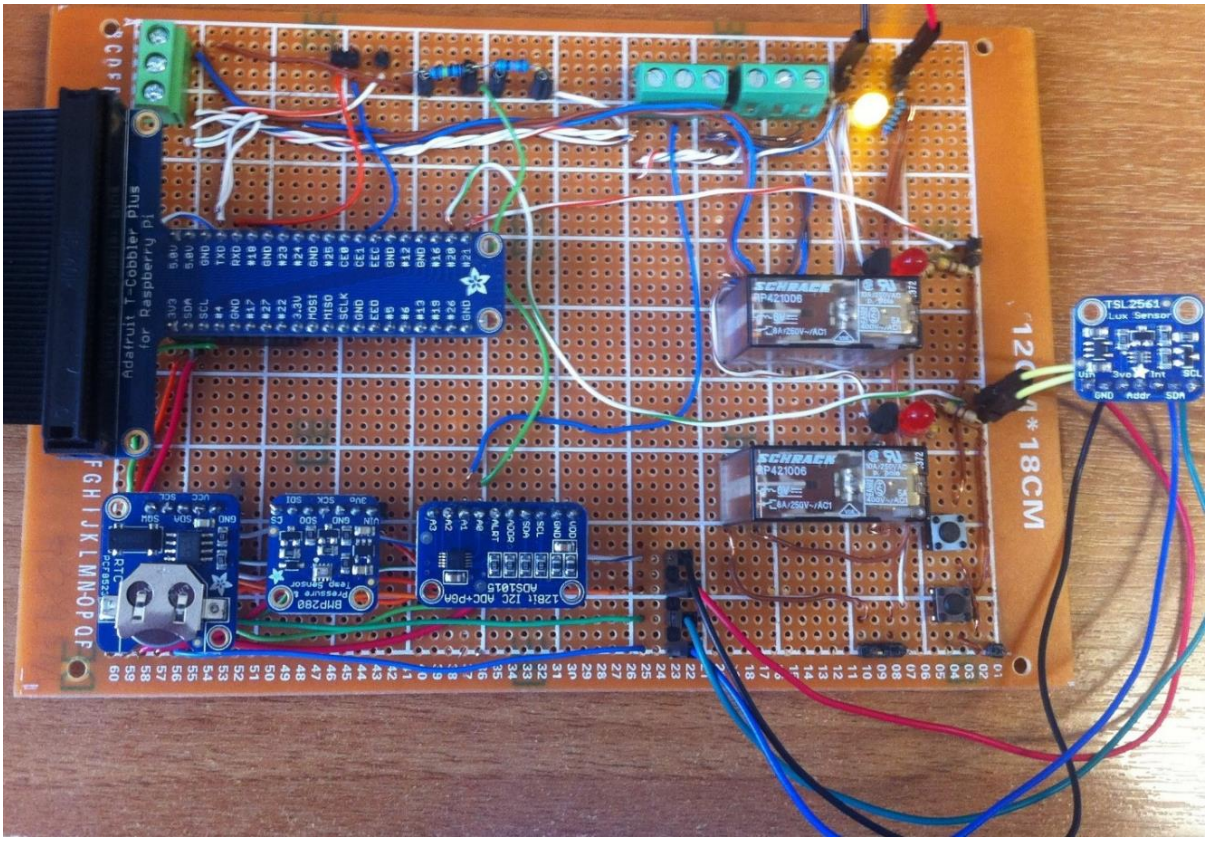


Figure 29. 2nd Custom board PCB layout

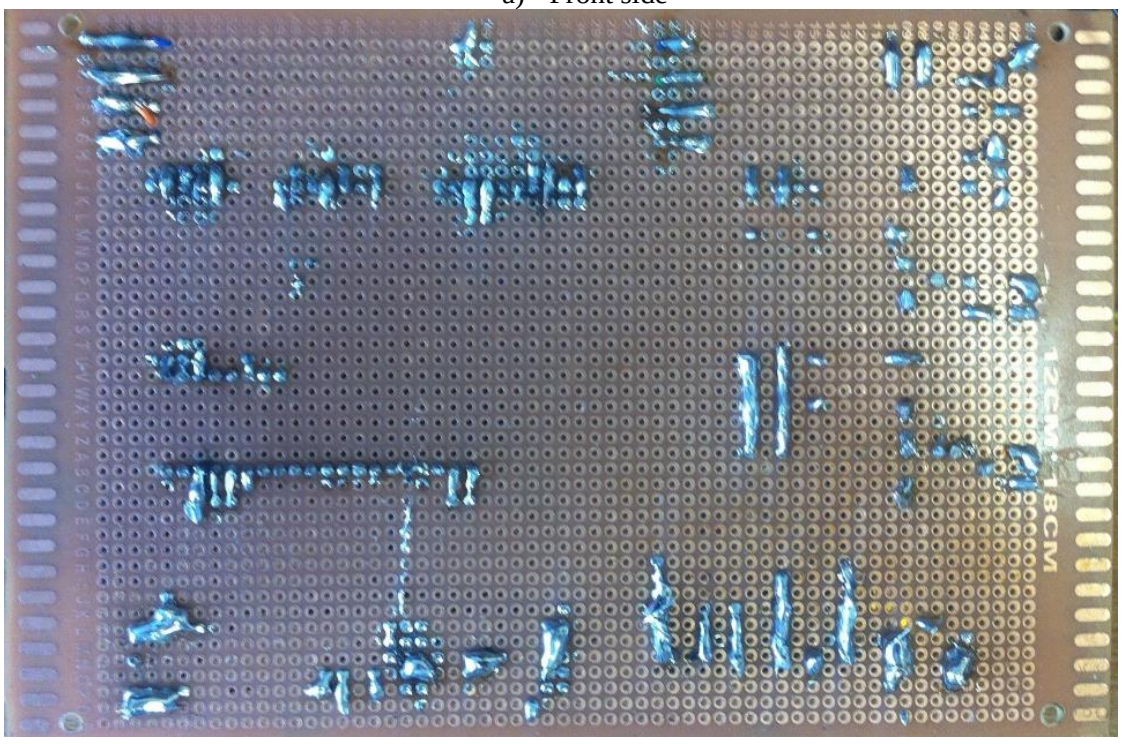
The fritzing application has many useful features and can be used to send designs to be etched to PCB.

3.1.12.4 Soldering all components on a perforated circuit board

We did not print a PCB, but instead we used a perforated circuit board, the main sensors were not soldered directly on the board but some straight PCB Sockets are used that allow to insert and remove components, for ease of testing and even component repurposing.



a) Front side



b) Back side

Figure 30. 2nd Custom soldered board

3.2. Operation of the Trap

Apart from the programs that handle reading values from the various sensors that are written in C or C++, the main scripts that are responsible for collecting data from all the sensors and sending them to the remote webserver are written in Python. The decision was made firstly because they would not have any performance gains if written in a more low-level language like C++ and secondly because it is much easier to maintain since we don't need to re-compile the programs when changes are made.

The scripts are scheduled to run as scheduled jobs (cronjob) and are setup to run every 2 minutes under the user account "pi", details can be found in the Appendix: A.7 Python scripts that collect sensor values and send them to the webserver.

These scripts handle all the logic for the operation of the trap and although originally were written as one script, they were later divided in two.

The first script "**`read_sennsor_data.py`**" handles the following functions:

- Create and store the current timestamp that will be used to identify and store data collected.
- Turn on and off relay switches use to power the LED lights, Anemometer and the GSM/GPRS module.
- Read values from the i2c bus sensors:
    - Humidity and Temperature
    - Barometric pressure and Temperature
    - Ambient light (Broadband, IR, Lux)
    - ADC (analog to digital converter) that reads: Battery voltage and Wind speed (anemometer)
    - Camera -Vision Sensor
- Creates a string that holds all sensor values and finally
- Saves the data to be saved in a specific location in the filesystem **`/trap/buffer`** with a filename that is the timestamp ending with the extension "`.json.dumps`"

The second script "**`upload_sensor_data.py`**" handles the following functions:

- Read all files in folder **`/trap/buffer`** with filename that ends in "`*.json.dumps`"
- Make a POST and attaches the JSON data stored in the file.
- Establish a secure connection with the web server (Https) and utilize a pre-shared key, unique to this trap. The web server accepts incoming connections only from authorized traps (using a correct hash)
- If data was sent successfully, move each file processed in the folder: **`/trap/buffer/processed`**

The above setup was initially contained in the main script but was later separated to handle cases where we would want to conserve battery life and not connect to the GSM/GPRS network every time we read data from the sensors. By separating them data can be collected for long periods of time and sent all at once together later when we establish an internet connection.

In the next figure, the operation of the automated trap is schematically described.

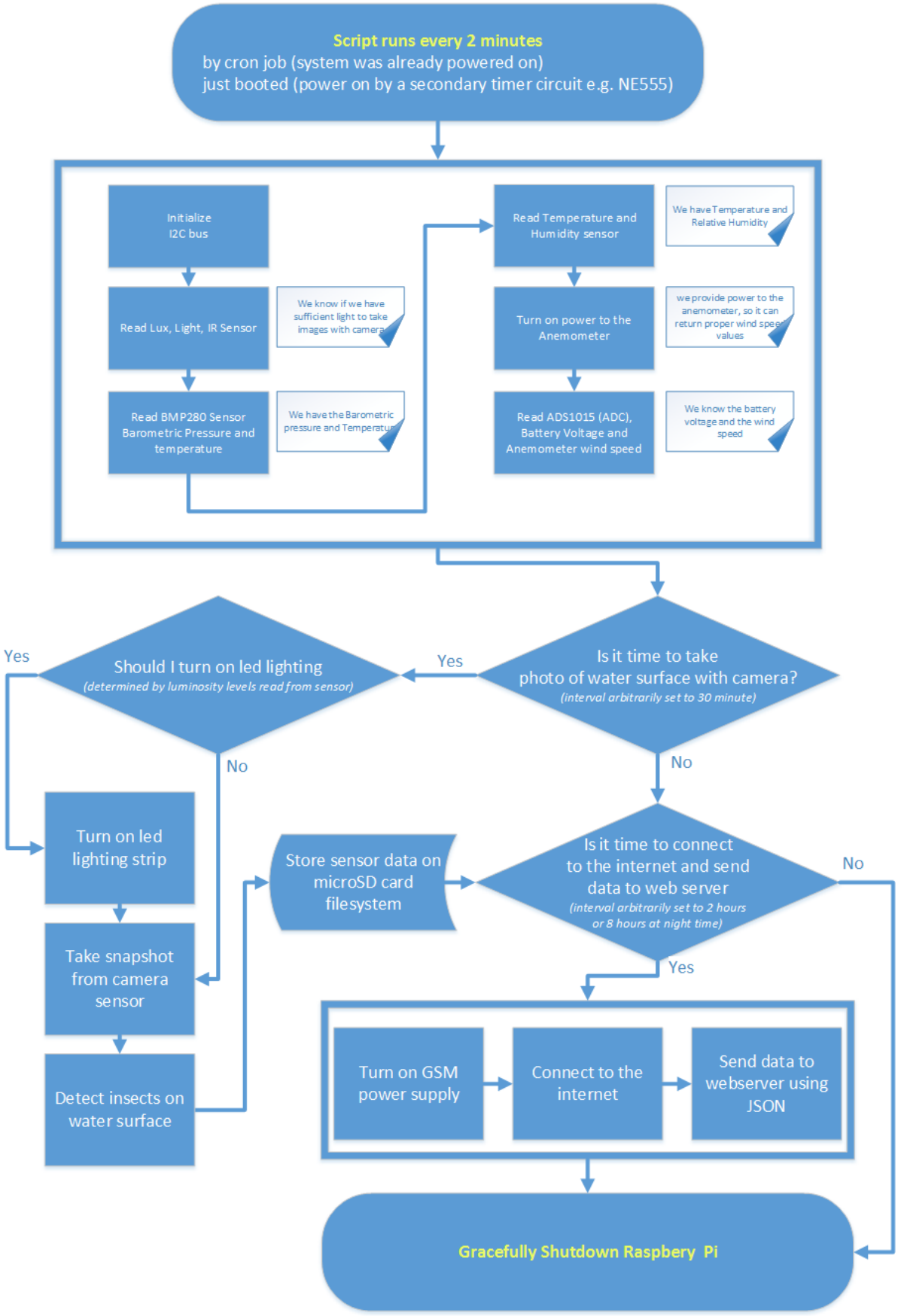


Figure 31. Operational Flowchart of the system

3.3. Methodologies and algorithms

3.3.1. Computer Vision

Computer vision is the process of transformation of data from a still or video camera into a decision or a new representation (Bradski & Kaehler, 2013). Although to humans, tasks like counting insects in a trap appear trivial, computers cannot perform them that easily. Special algorithms have been developed over time that aid in computer vision tasks, and one of the most well-known libraries used, is Open Source Computer Vision Library (OpenCV) that is written in C and C++ and runs under Linux, Windows and Mac OS X operating systems. The OpenCV library was initially developed by a team of Intel developers as an unofficial open source Intel project but now is supported by the open source community.

Although OpenCV is released as open source, it's licensing scheme is such that it encourages the creation of applications that can be turned into commercial products. On the other hand some functionalities cannot be used in commercial applications without paying a license fee to the patent holder such examples are the SHIFT and SURF algorithms that are patented in the United States (Brahmbhatt, 2013).

Raspbian OS comes with OpenCV pre-installed but it's not the latest version, so following the guide from pklab.net (PkLab.net, 2017), we uninstalled the old version and after installing all the prerequisite compilation tools, Raspbian Jessy modules and libraries we downloaded the latest version from the source forge repository, and successfully compiled OpenCV from source files. Afterwards we run a simple C++ script, verifying we could read from the camera sensor and perform basic OpenCV functions.

In this project OpenCV and machine vision algorithms played a key role in the proper operation of the key functionality of the trap trying to create code that executes as fast and efficient as possible utilizing the limited Raspberry Pi resources.

When working with images using OpenCV, Matlab, or any other program, we are working with Matrixes. Each image when captured from a digital camera or scanned, when loaded in memory is represented as a matrix object. When working with a single channel (grayscale images) the matrix has as many rows and columns as the pixels in the original image, and each pixel has a value representing the light intensity (or relative luminance) at that specific pixel position.

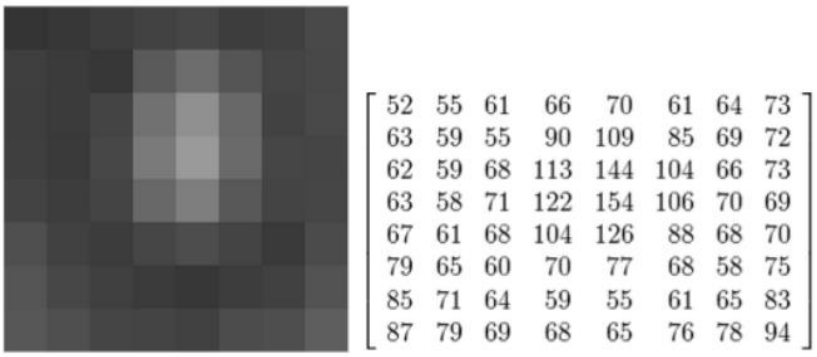


Figure 32. An image and its light intensity representation in a matrix, image taken from *(Hong, 2016)*

Color images are represented as multi-dimensional matrices, each dimension represents light intensity of that specific color channel, in some cases another alpha channel is used to represent image transparency.

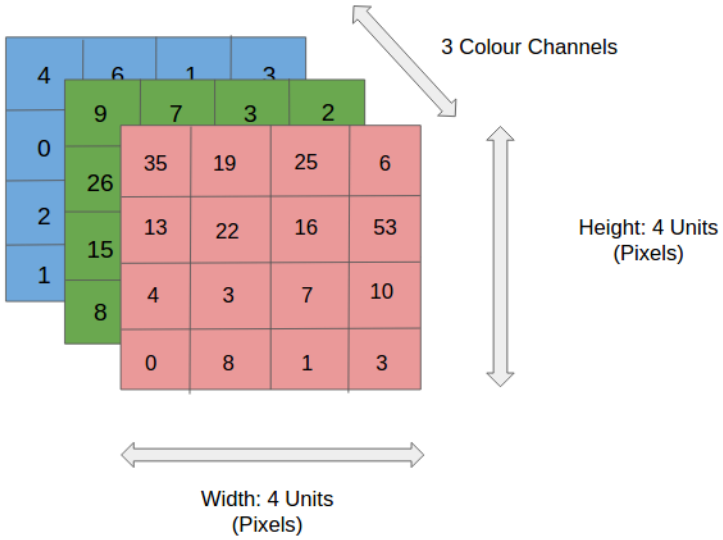


Figure 33. A color image multi-dimensional matrix representation, image taken from *("Convolutional neural networks cnns," 2016)*

Apart from the above concepts mentioned, another thing to note is the datatype, i.e. numbers used to represent pixel intensity. In the two above examples, (Figure 32 and Figure 33) 8-bit unsigned integers are used with values ranging from 0 to 255. When 255 intensity levels are not enough to describe more complex images, we can capture or convert images to other data types. Available datatypes in OpenCV are shown in the following table:

Table 1. **A Mapping of Type to Numbers in OpenCV** (taken from (Hang, 2012)**)**

	C1	C2	C3	C4
CV_8U	0	8	16	24
CV_8S	1	9	17	25
CV_16U	2	10	18	26
CV_16S	3	11	19	27
CV_32S	4	12	20	28
CV_32F	5	13	21	29
CV_64F	6	14	22	30

Table values represent the Matrix datatype and precision used internally in OpenCV,

- Number of channels in image is represented by the C1, C2, C3, C4, columns.
- The numbers used to represent pixel intensity / relative luminance are:
    - CV_8U has 8-bit unsigned chars, values ranging from 0 ~ 255
    - CV_8S has 8-bit chars, values ranging from -128 ~ 1228
    - CV_16U has 16-bit unsigned short, values ranging from 0 ~ 65535
    - CV_16S has 16-bit short, values ranging from -32768 ~ 32767
    - CV_32S has 32-bit integer, values ranging from -2147483648 ~ 2147483647
    - CV_32F has 32-bit float, values ranging from $-1.18\ 10^{-38}$ ~ $3.4\ 10^{38}$
    - CV_64F has 64-bit double precision

When we apply filters, or do any image manipulation we manipulate pixel values stored in these matrixes.

3.3.2. Convolution matrix operations

In image processing many operations like blurring, sharpening, embossing, edge detection, etc, are performed by using small matrices usually referred to as masks, convolution matrices or Kernels. They are a fixed size array of numerical coefficients, with an anchor point typically located at the center or the matrix. Matrix convolution is usually denoted by * but is not a traditional matrix multiplication, what the output of a convolution will be, is determined by the actual Kernel used ("Kernel," 2012).

Convolution matrices we use have two key features: they are shift-invariant, and they are linear. *Shift-invariant* means that we perform the same operation at every point in the image. *Linear* means that this operation is linear, that is, we replace every pixel with a linear combination of its local neighbors, weighted by the Kernel. These two properties make these operations very simple; it's simpler if we do the same thing everywhere, and linear operations are always the simplest ones (Jacobs, 2005).

3.3.3. Noise in digital images

All images produced by digital cameras have noise, that creates problems to computer vision algorithms. The origin of noise in digital cameras can be: the physical spacing between adjacent sensor cells (sampling pitch) where smaller pitch means that individual sensors occupy smaller area and cannot accumulate as many photons, chip size that is analogous to sensor cell (bigger chip size is preferable), analog gain of the ADC is like the film camera ISO setting but in practice higher analog gain settings can also amplify sensor noise (Szeliski, 2010).

3.3.4. Applying filters to digital images and Blurring

There are two broad categories of image enhancement techniques: applying filters to images on the spatial domain (direct manipulation of image pixels), and applying filters on the frequency domain (manipulating the Fourier transform of an image). The filters we are going to use belong to the spatial domain techniques.

Smoothing also called blurring is a commonly used computer vision operation, that is applied on an image, to remove noise, make it more appealing or even highlight interesting details. Using different kernels can remove different types of noise or achieve other effects (Szeliski, 2010), (Bradski & Kaehler, 2013).

One important continuous function used for image smoothing it the Gaussian filter.

A one-dimensional Gaussian is also known as a Normal distribution: $G_{1D}(x,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$, where μ is the mean value (and gives the location of the peak of the function) and parameter σ is the variance (and controls how wide the peak is). As σ gets smaller the peak becomes narrower.

When Gaussian is used as a filter, each pixel it is applied upon is replaced, with the average of its neighbors, where nearby pixels play an equal role on the average and the further away they are they influence less or not at all (depending on the Kernel size). When we use Gaussian for smoothing, μ is set to zero, because we want the pixel we apply it on to have the biggest effect on its new smoothed value.

$G_0(x,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{x^2}{2\sigma^2})$, parameter σ is used to allow us to control how much we smooth the image, the bigger it is the more we smooth the image.

If we wanted to create a discrete filter that looks like a Gaussian, we can do this by evaluating the Gaussian at discrete locations for example for x=[-3,-2,-1,0,1,2,3] two things should be noted, since the filter is used for smoothing and we do not want to alter the image brightness, the sum of all filter elements must add up to 1.

A two-dimensional Gaussian is: $G_{2D}(x,y,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(x^2+y^2)}{2\sigma^2})$, parameter σ is the variance.

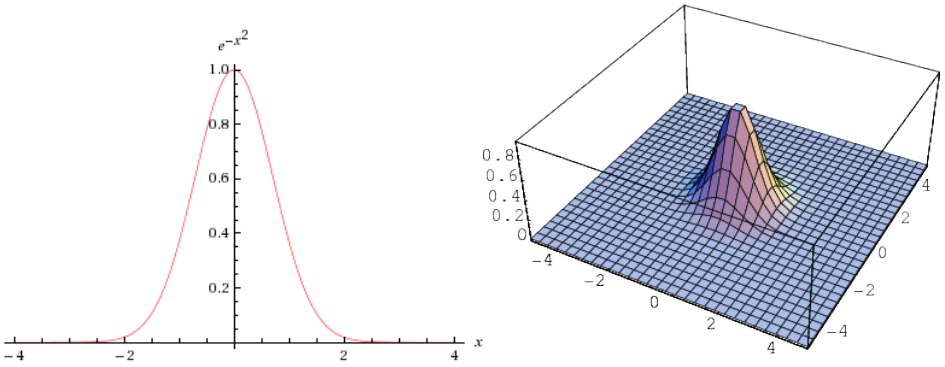


Figure 34. Plot of Gaussian distributions, one-dimensional and two dimensional, image taken from *(Weisstein, 2012)*

If a 2D Gaussian filter Kernel is closely examined one can notice two properties: the 2d matric is circularly symmetric and can be separated into two one dimensional filters that can be convoluted to achieve the same result as we would by using the 2D filter but making fewer calculations.

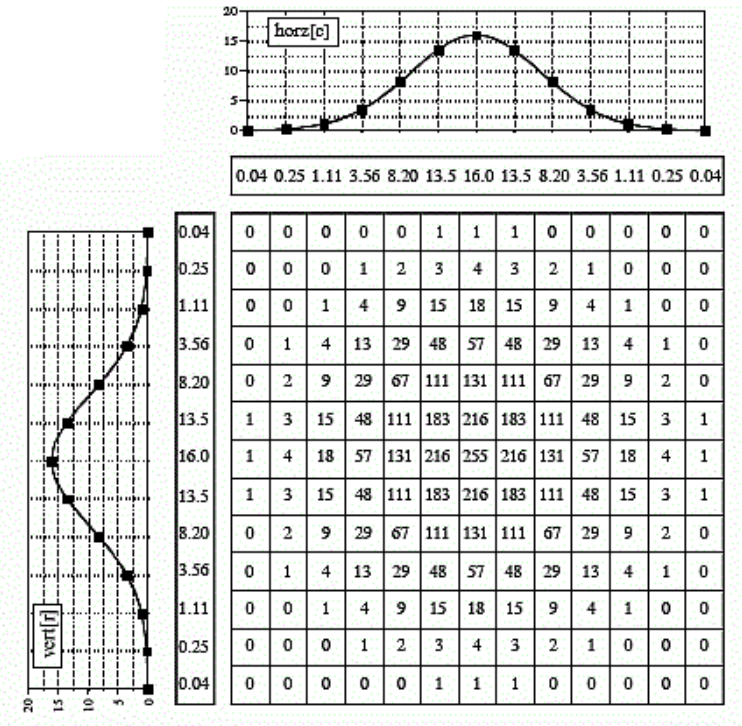


Figure 35. The Gaussian is circularly symmetric and separable, image taken from *(Smith, 2002)*

3.3.5. Image Segmentation

Image segmentation is the process of partitioning - decomposing an image into multiple parts to simplify and/or change it into something that will make it easier to be later analyzed. Segmentation is one of the first steps in image analysis and in general, autonomous segmentation is one of the most difficult tasks in digital image processing.

Several methods have been developed over the past decades to perform image segmentation that tackle specific problems that were not sufficiently addressed by other preexisting methods or were not as efficient. Characteristics that are used to perform segmentation can be computed properties such as gray level, color, texture, depth, intensity or motion (Lucas, 2010). Selecting the appropriate method for performing image segmentation can be a difficult decision.

Automatic thresholding of grayscale images has been used for pattern recognition, optical character recognition and other fields of computer vision but it works best when the image histogram has a bimodal distribution (Yang, Shen, Long, & Chen, 2012) c

During the test and in the final design, the following machine vision methods have been used to aid in the image decomposition: Gaussian Blurring, Hough circle transform, custom thresholding, image binarization using Otsu's method thresholding, finding blob contour and area.

3.3.6. Image segmentation using histograms

With histogram based segmentation algorithms, an image is partitioned into regions that have similarities using some predefined criteria. An image histogram can be generated and displayed as a chart, in the horizontal axis listing all possible light intensity values a pixel could have (referred to as bins) and in the vertical axis shows the pixel count of each light intensity value (how many pixels belong to that bin). When working with grayscale images the horizontal axis has 256 values and in the following figure, a sample histogram is shown.

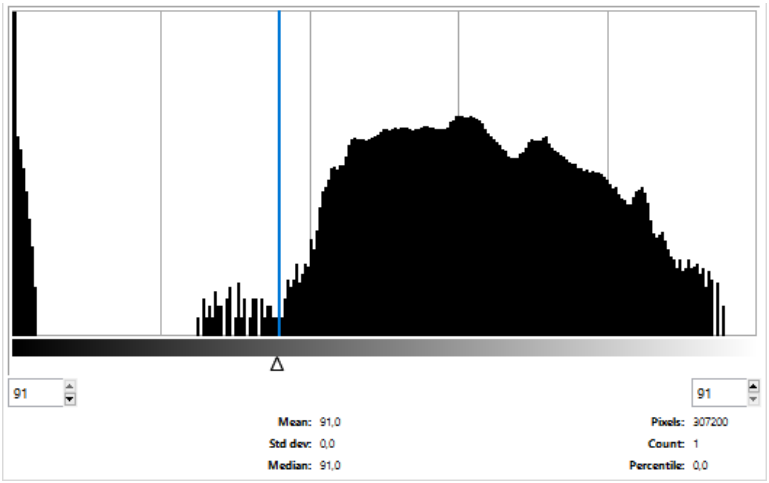


Figure 36. Image histogram (computed with gimp)

3.2.4.7. Thresholding

This method attempts to partition an image by dividing pixels into two groups, background pixels and pixels that represent an object. The resulting image has only two levels.

It is a generally easy process but for it to be effective it should preserve logical and semantic content. There are two types of thresholding algorithms:

- Global thresholding algorithms: A single numeric threshold is chosen for the whole image and by comparing its value to each pixel in the image a value is assigned (e.g. white for background and black for foreground objects). Images that are the best candidates for thresholding are those with a bimodal image histogram (Devi, 2006) (Lucas, 2010) (Chaubey, 2016) That is an ideal case where the histogram has a deep and sharp valley between two peaks representing objects and background (Otsu, 1979).
- Local or adaptive thresholding algorithms: Different threshold values are used for different local areas.

3.3.7. Thresholding using Otsu's method

Otsu's thresholding method belongs to global thresholding algorithms, and is an iterative process, that selects a threshold value where the sum of foreground and background spreads is at its minimum.

Let the pixels of a given picture be represented in L gray levels [1,2,3…L]. The number of pixels at level i is denoted as n_i and the total number of pixels by $N = n_1 + n_2 + \ldots + n_L$.

For simplification, the gray-level histogram is normalized and regarded as a probability distribution.

$$p_i = \frac{n_i}{N},\ p_i \geq 0, \sum_{i=1}^{L} p_i = 1 \quad (1)$$

If we divide all pixels into two classes C_0 and C_1 (one of them being the background and one the objects) by a threshold at level k; C_0 contains pixels with levels [1,…,k] and C_1 contains pixels with levels [k+1, …, L]. Then the probabilities of a class occurrence and the class mean levels, respectively, are given by

probabilities of a class occurrence

$$\omega_0 = \Pr(C_0) = \sum_{i=1}^{k} p_i = \omega_{(k)} \quad (2)$$

$$\omega_1 = \Pr(C_1) = \sum_{i=k+1}^{L} p_i = 1 - \omega_{(k)} \quad (3)$$

and class mean levels

$$\mu_0 = \sum_{i=1}^{k} i\ \Pr(i|C_0) = \sum_{i=1}^{k} i\, p_i/\omega_0 = \mu(k)/\omega(k) \quad (4)$$

$$\mu_1 = \sum_{i=k+1}^{L} i\ \Pr(i|C_1) = \sum_{i=k+1}^{L} i\, p_i/\omega_1 = \frac{\mu_T - \mu(k)}{1-\omega(k)} \quad (5)$$

Where

Zeroth-order cumulative moments $\omega_{(k)} = \sum_{i=1}^{k} p_i$ (6)

First-order cumulative moments $\mu(k) = \sum_{i=1}^{k} ip_i$ (7)

of the histogram, up to k-th level.

$$\mu_T = \mu(L) = \sum_{i=1}^{L} ip_i \text{ (5)} \quad (8)$$

For any choice of k we can verify that

$$\omega_0\mu_0 + \omega_1\mu_1 = \mu_T,\ \omega_0 + \omega_1 = 1 \quad (9)$$

The class variances are given by

$$\sigma_0^2 = \sum_{i=1}^{k}(i-\mu_0)^2\, P_r(i|C_0) = \sum_{i=1}^{k}(i-\mu_0)^2\, p_i/\omega_0 \quad (10)$$

$$\sigma_1^2 = \sum_{i=k+1}^{L}(i-\mu_1)^2\, P_r(i|C_1) = \sum_{i=k+1}^{L}(i-\mu_1)^2\, p_i/\omega_1 \quad (11)$$

To evaluate how good the selected threshold at level k is, a measure of class separability is introduced.

$$\lambda = \frac{\sigma_B^2}{\sigma_w^2}\ ,\quad \kappa = \frac{\sigma_T^2}{\sigma_w^2}\ ,\quad \eta = \frac{\sigma_B^2}{\sigma_T^2} \quad (12)$$

Where

within class variance $\sigma_W^2 = \omega_0\sigma_0^2 + \omega_1\sigma_1^2$ (13)

between class variance $\sigma_B^2 = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2$

$$= \omega_0\omega_1(\mu_1 - \mu_0)^2 \quad (14)$$

And due to (9)

Total variance $\sigma_T^2 = \sum_{i=1}^{L}(i-\mu_T)^2 p_i$ (15)

The problem is reduced to an optimization problem to iteratively search for the threshold k that maximizes one of the object functions (criterion measures in (12)), the criterion maximizing $\boldsymbol{\lambda}$,$\boldsymbol{\kappa}$ and $\boldsymbol{\eta}$ are equivalent to one another, $\kappa=\lambda+1$ and $\eta=\lambda/(\lambda+1)$ in terms of λ, because relation (16) always holds (Otsu, 1979)

$$\sigma_W^2 + \sigma_B^2 = \sigma_T^2 \quad (16)$$

and although σ_W^2, σ_B^2 are functions of the threshold level k, σ_T^2 is independent of k.

the optimal threshold k^*, maximizes η or equivalently maximizes σ_B^2 can be selected by the relations (6) and (7) or (2)-(5)

$$\eta(k) = \frac{\sigma_B^2(k)}{\sigma_T^2} \quad (17)$$

$$\sigma_B^2(k^*) = \frac{[\mu_T\,\omega(k) - \mu(k)]^2}{\omega(k)\,[1-\omega(k)]} \quad (18)$$

And the optimal threshold k^* is

$$\sigma_B^2(k^*) = \max_{1 \le k < L} \sigma_B^2(k) \quad (19)$$

So, the problem is restricted to the effective range of the gray-level histogram.

$$S^*=\{k;\ \omega_0\omega_1 = \omega(k)[1-\omega(k)] > 0,\ \text{or } 0< \omega(k)<1\}$$

3.3.8. Algorithm data flow

Based on the algorithms described in the previous paragraphs we have developed a procedure which is depicted in detail in Figure 38, - Figure 41 and that allows us to have an estimate of the number of insect inside the trap. The computer vision sequence of image transformations performed are shown in the following date flow chart:

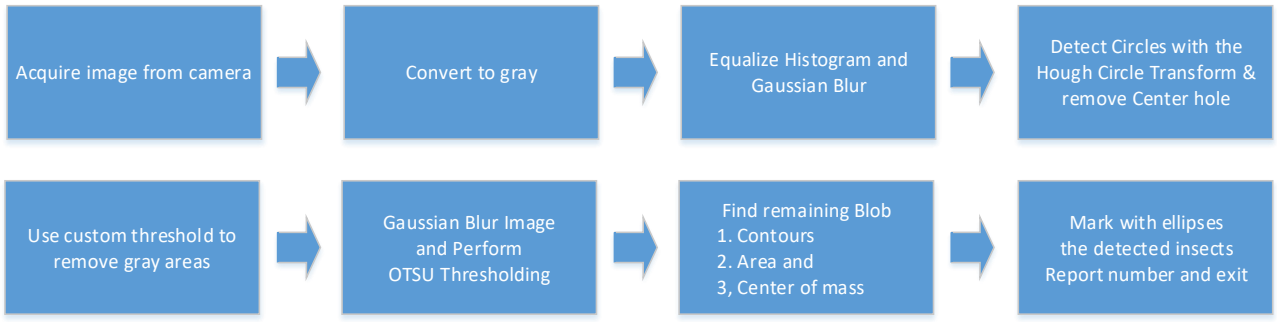


Figure 37. Machine vision algorithm operations in sequence

1. Acquire an image from the camera,
2. Convert it to grayscale
3. Equalize Histogram on grayscale image
   Gaussian Blur to reduce the noise and prepare it for the Hough circle transform
4. Find a circle that has a radius between 90-130 pixels (center hole) using the Hough circle transform
5. Create a mask to focus on Region of Interest
6. Use mask created above to remove the outer and center regions leaving only the liquid surface
7. Use a custom threshold to remove gray areas that are not of interest
8. Gaussian blur to reduce the noise and prepare the image
   perform OTSU thresholding
9. Find all connected blobs,

   Drop any blobs with an area higher that a certain threshold,

10. Save final image in /trap/grabbed/ and show a number on the terminal

Let’s follow through these steps in more detail:

Steps 1-6, are performed to discerning the center hole of the trap, which is expected to have dark areas that would erroneously be picked up as insects.

1. Acquire an image from the camera,

```
VideoCapture cap(0);
cap >> frame;
```

2. Convert it to grayscale

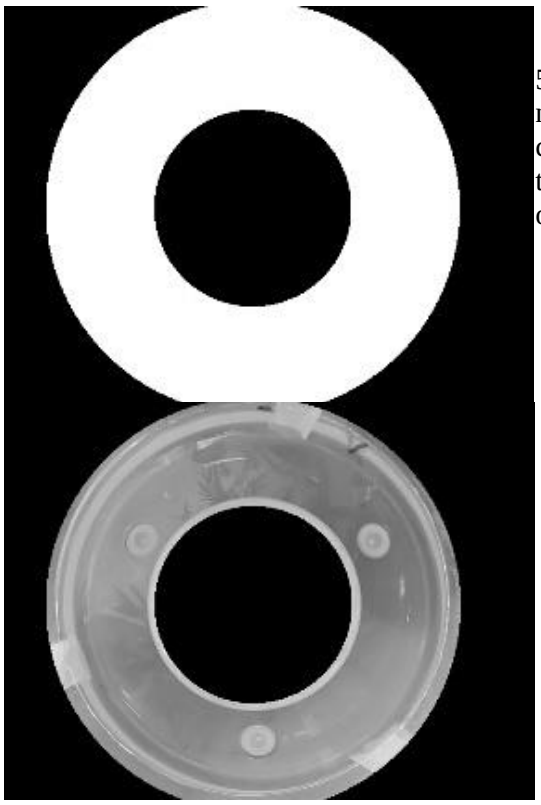
```
cvtColor( frame, gray_image, CV_BGR2GRAY );
```

3. Equalize Histogram on grayscale image
   Gaussian Blur to prepare it for the Hough circle transform

```
cv::equalizeHist(gray_image,histogramEqualiz
ed);
GaussianBlur(histogramEqualized, blured,
Size(9,9),2,2);
```

4. Find a circle that has a radius between 90-130 pixels (center hole) using the Hough circle transform

```
HoughCircles( blured, circles,
CV_HOUGH_GRADIENT, 1, gray_image.rows/8, 20
0, 100,90 , 130 );
```



5. Create a mask to be used to remove areas that we do not need, use a black foundation, draw a filled circle using the center of the hole detected on step 4 and diameter the height of the image (white color) and after that paint the circle detected on step 4 (black color).

```
//create mask for gray image
circle(liquidMaskg,center,10+gray_image.rows
/2, Scalar(255,255,255), -1, 8, 0);
circle(liquidMaskg,center,radius,Scalar(0,0,
0), -1, 1, 0);
```

6. Use mask created above to remove the outer and center regions leaving only the liquid surface

```
gray_image.copyTo(img_maskedg,liquidMaskg);
```

Figure 38. Machine vision, Steps 1-6, (focus on region of interest)

The output image of step 6, is a masked grayscale image that contains only the region of interest we want to focus on[2].

Step 7-9: Having already removed the center hole, the darkest spots on the image should be only insects on the liquid surface. One way to isolate just them, is to do a custom thresholding operation, eliminating any pixels with intensity higher that a certain predefined threshold – setting their intensity to 255 (white).

If we examine the grayscale image histogram, we can see that a most of the image contains gray pixels with intensity over 100.

[2] Note: Since we are making a custom trap and we know the position of all components, another approach could be to measure the trap hole position once and hard code the center and diameter in our C++ code to remove it without doing any Hough circle transform calculations.

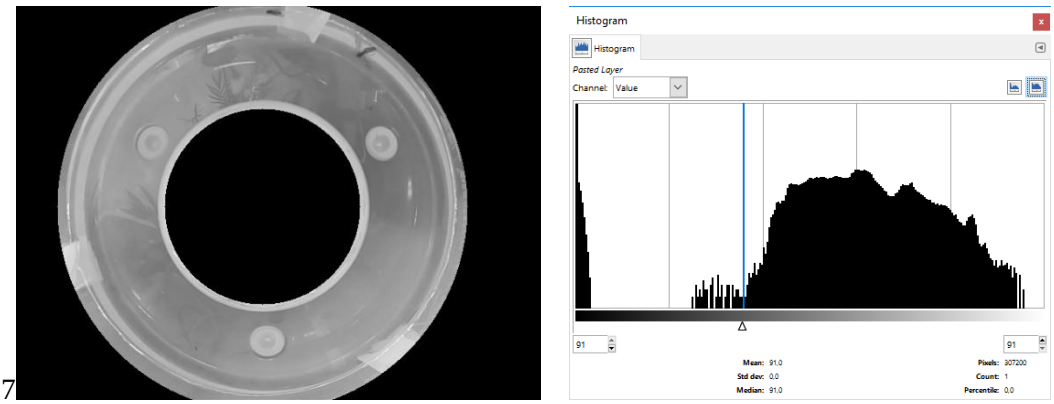
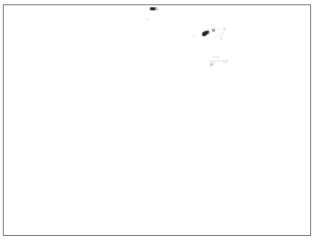
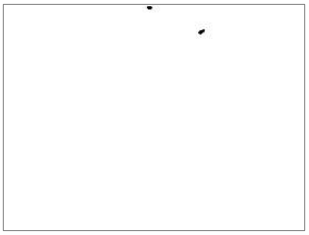
7 

Figure 39. Image after processing of step 6 and corresponding image histogram (computed with gimp)



7. use a custom threshold to remove all gray areas that have intensity greater thatn 100.

```
int threshold_lux_norm =100;
for(int y=0;y<grayHole.rows;y++) {
 for(int x=0;x<grayHole.cols;x++) {
   //if current pixel value > that threshold value
   If((int)grayHole.at<uchar>(y,x)>threshold_lux_norm)
   {
         erase_grayishAreas.at<uchar>(y,x)=255;
   }
 }
}
```



8. Gaussian blur to reduce the noise and prepare the image
   Perform OTSU thresholding

```
GaussianBlur(grayHole, blured, Size(9,9),2,2);
threshold(blured,im_bw,0,255,CV_THRESH_BINARY|CV_THRESH_OTSU);
```

Figure 40. Machine vision, Step 7-9, (isolate insects – low luminosity pixels)

After step 7, the image contains only a few blobs that should represent insects. We perform Gaussian blur operation to reduce possible image noise and then perform binary thresholding using Otsu's method.

Now we need to identify and determine if the remaining blobs are indeed insects.

At Step 9, we must count the dark regions (blobs) on the image and try to find their contours and area. Insects are expected wo have a small area so we discard blobs with area bigger that a custom threshold.

Listing 1. Machine vision, Step 9, (find blob contours, discard if area too big)

Step 9. Find contours of all objects in the image

```
vector<vector<Point> > contours;
findContours(im_bw, contours, RETR_LIST, CHAIN_APPROX_NONE);

// Get the moments
vector<Moments> mu(contours.size() );
for (int i = 0; i < contours.size(); i++) {
   mu[i] = moments( contours[i], false );
}
//  Get the mass centers:
vector<Point2f> mc( contours.size() );
for( int i = 0; i < contours.size(); i++ )  {
   mc[i] = Point2f( mu[i].m10/mu[i].m00 , mu[i].m01/mu[i].m00 );
}

Mat cimage = Mat::zeros(im_bw.size(), CV_8UC3);
Mat cimageOriginal; frame.copyTo(cimageOriginal);
int insectsFound=0;
for(size_t i = 0; i < contours.size(); i++) {
   size_t count = contours[i].size();
   cout << " Contour " << i << "  Area: " << contourArea(contours[i]) << "
Center : (" << cvRound(mc[i].x) << ","<< cvRound(mc[i].y) << ")"<< std::endl;
   // determine if area of contour is too big... it is probably a shadow.
   if (contourArea(contours[i]) > 400) {
      cout << "Skipping Too big " <<endl;    continue;
   }

   if( count < 20 )
      continue;
   Mat pointsf;
   Mat(contours[i]).convertTo(pointsf, CV_32F);
   RotatedRect box = fitEllipse(pointsf);
   if( MAX(box.size.width, box.size.height) > MIN(box.size.width,
box.size.height)*30 )
      continue;

   drawContours(cimage, contours, (int)i, Scalar::all(255), 1, 8);
   drawContours(cimageOriginal, contours, (int)i, Scalar::all(255), 1, 8);
   ellipse(cimage, box, Scalar(0,0,255), 1, LINE_AA);
   ellipse(cimageOriginal, box, Scalar(0,0,255), 1, LINE_AA);
   ellipse(cimage, box.center, box.size*0.5f, box.angle, 0, 360,
Scalar(0,255,255), 1, LINE_AA);
   ellipse(cimageOriginal, box.center, box.size*0.5f, box.angle, 0, 360,
Scalar(0,255,255), 1, LINE_AA);
   Point2f vtx[4];
   box.points(vtx);
   insectsFound++;
   for( int j = 0; j < 4; j++ ) {
      line(cimage, vtx[j], vtx[(j+1)%4], Scalar(0,255,0), 1, LINE_AA);
      line(cimageOriginal, vtx[j], vtx[(j+1)%4], Scalar(0,255,0), 1, LINE_AA);
   }
}
```

Figure 41. Machine vision, Step 9, (original Image with bright green contours drawn where the algorithm found insects)

Finally, the program saves the image to a disk file and prints out a report[3]

Listing 2. Machine vision, Step 10, screen output to the screen

Step 10. Output from the script

```
# /trap/LightsON.py;   /trap/cpp/grab_removeHole_histEq_OtsuThresh;
  /trap/LightsOFF.py;

LED 21 ON (LED Strip - Anemometer)

timestamp: 2017_06_23__08-14-36

Start grabbing
Will try to find Circles, in order to detect the center hole
Circle1: center : [306, 250] radius : 129

Contour 0  Area: 215    Center : (499,244)
Contour 1  Area: 61     Center : (500,222)
Contour 2  Area: 306081 Center : (320,240)    Skipping Too big
Found 2

Closing the camera
bye!
```

[3] During the development phase, a lot of problems were faced and the algorithm was changed many times because it did not always work. Some examples of cases where the code reported wrong number of insects are mentioned at §4.1.

3.4. Web Server

The proposed solution is to have dedicate a server that will handle all incoming communication from the McPhail traps, store the data in a database and finally handle the display of collected data.

The mission of the server is to:

- Allow secure connections to be established with authorized traps and data collected are authentic. This is accomplished by using HTTS connections (SSL Encryption) and all data is sent via POST using a Pre-shared secret hash/key.
- Store data to a local MySQL database
- Display data to anonymous users (a limited result set – for example last 500 sensor data values)
- Display data to authorized users (users can query database to see all historical data stored)
- Display and edit data to selected administrators (these users have permission to edit data as well)
- Backup the database (both locally and at a remote location)

Every trap that is deployed, is shown using a marker and added to the google maps view. Clicking on a marker updates the web interface showing details about the sensors (Figure 48. Screenshot of the web server interface). The web interface is responsive and is usable even on smart phones with smaller screen size, the size and position of charts adjust to fit on the screen (Figure 49. a) Responsive design adapts to mobile device's screen, b) graph and images) more details are in the section ***3.4.4. Apache HTTP and web User*** Interface.

When deployed in big numbers the automated McPhail traps are expected to collect a considerable amount of sensor data that needs to be stored on the remote server.

In the following paragraphs, we are going to expand on the server components that web interface is divided into two sections, on the top section a geographical map is displayed with

Selected traps from each geographical location could be set as open and have them release their data to the public but other than that but only the last 500 values are exposed.

The sensors currently connected to the trap give us access to the following parameters: Temperature, Humidity, Barometric Pressure, Battery voltage, Wind Speed, Light Lux, IR, Broad band, Number of insects detected, Image of trap interior.

3.4.1. Creating the Linux server

During development, we wanted to have a scalable server, that could be accessible from anywhere in the internet. Many service providers give out free evaluation period for their products, the Microsoft Azure platform was found to be the easiest to setup and is seamlessly integrated with the institutional account of TEI of Crete. To create or use the Azure platform we visit the webpage: http://portal.azure.com and we are required to login.

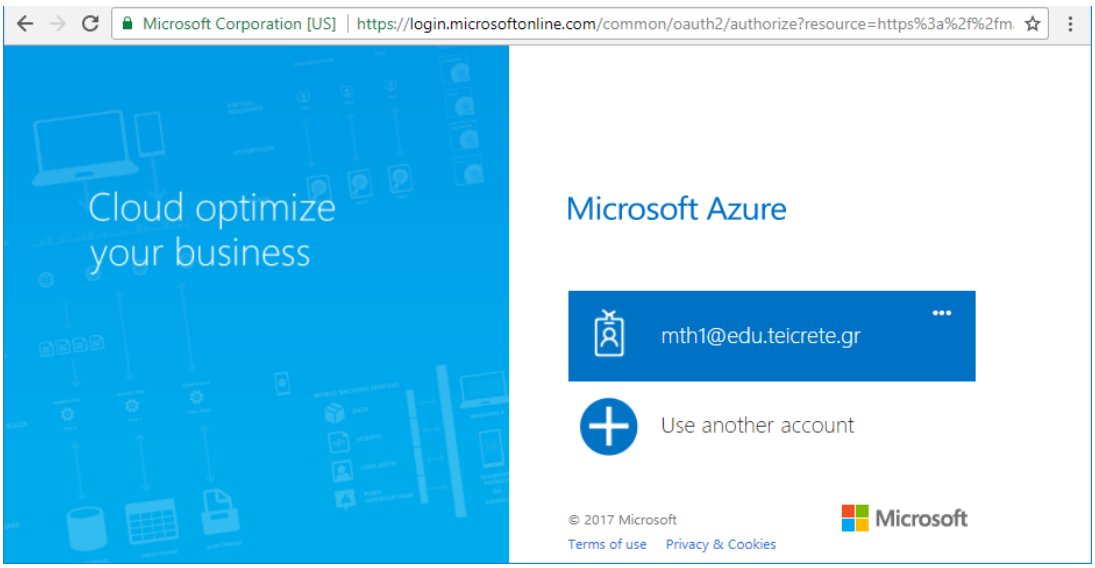


Figure 42. Login screen of the cloud platform Microsoft Azure

Once we type in our email the system redirects us to the Central Authentication System of the TEI of Crete where we enter our account password.

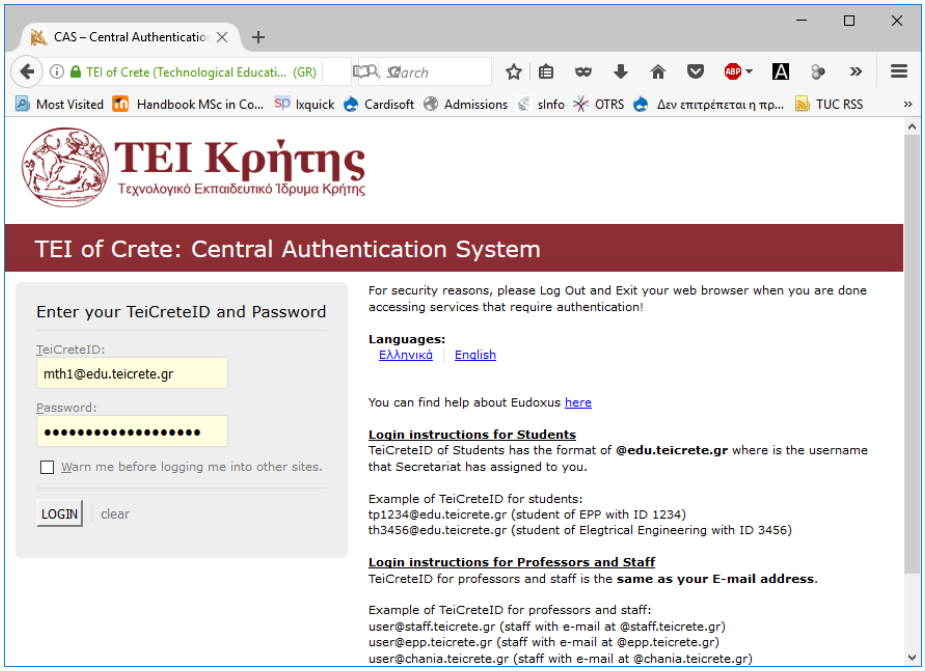


Figure 43. Login screen of the TEI of Crete Central Authentication System

After we enter our credentials we are redirected back to the Azure portal website.

As mentioned before this cloud service is from Microsoft but there are other cloud server providers like Amazon, Google etc. All the above services are free for evaluation purposes and either have a time limit, or resource limit after which you are required to buy a service. All cloud server providers, require that even users evaluating their services verify their identity by entering credit card information. As stated in their End User License Agreement the card will not be charged, but this step is required to verify they are humans and not Bots.

The fist screen is like a dashboard that displays the resources, virtual machines that are active and gives a easy way to create new virtual machines.

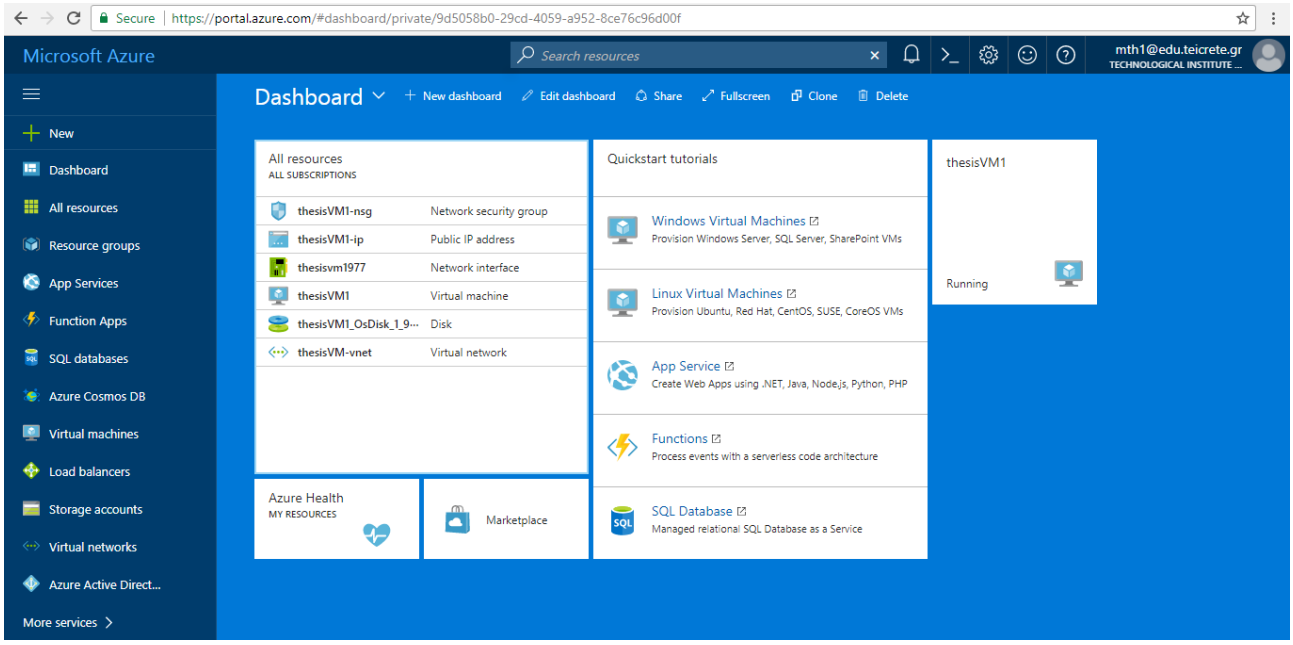


Figure 44. Microsoft Azure dashboard, quick access to all main functions and resources

After creating and account with the Microsoft Azure, a decision was made to setup an Ubuntu 16.04LTS Linux server. The server is provisioned with 1.75GB Ram, 1 Processor, 30GB Hard disk and one 1Gbps network interface, all of which are easily upgraded.

For the Ubuntu machine to act as a web server apart from the basic functionality, some additional services are required, Apache HTTPd and MySQL database server are going to handle the communication and store data received to be later analyzed and displayed.

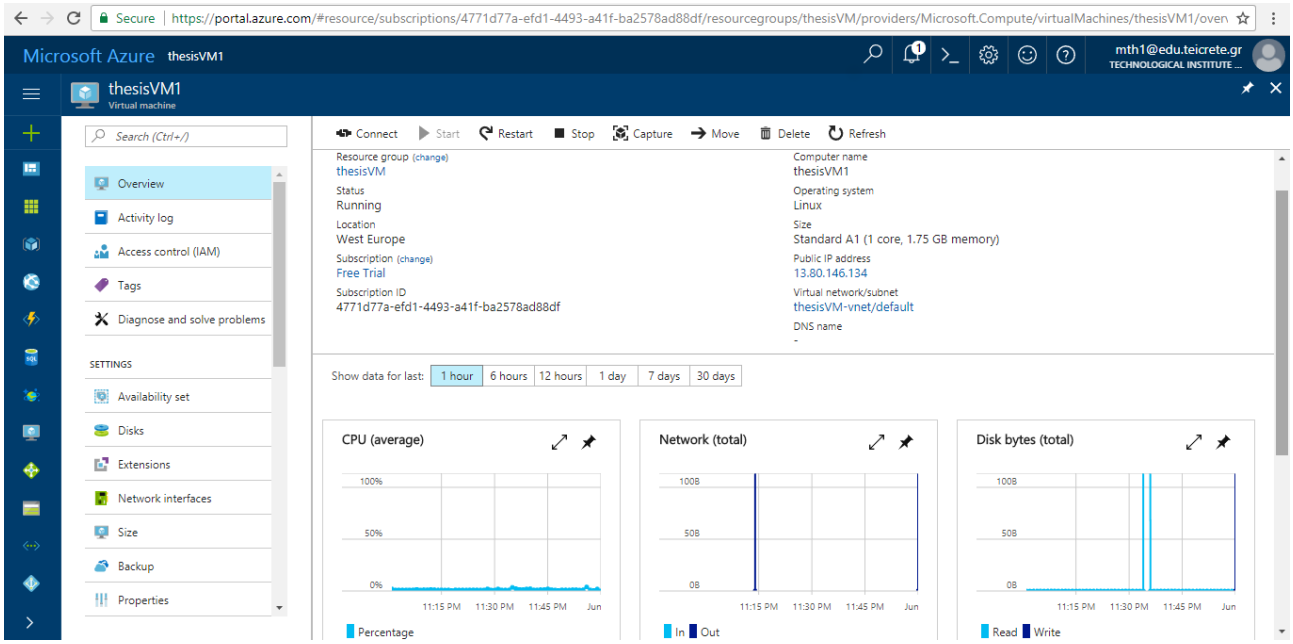


Figure 45. Details and monitoring graphs for the Linux webserver

3.4.2. Issue an SSL certificate

Typically, SSL Certificates are used to establish encrypted connections between a user's browser and a server. The SSL connection protects sensitive data, such as credentials sent during login and even data exchanged during each visit, which is called a session, from being intercepted from non-authorized parties.

Ssl.com issues SSL certificates that are valid for 90 days, to try out their service. Free SSL certificates can be issued by following the link https://www.ssl.com/certificates/free/

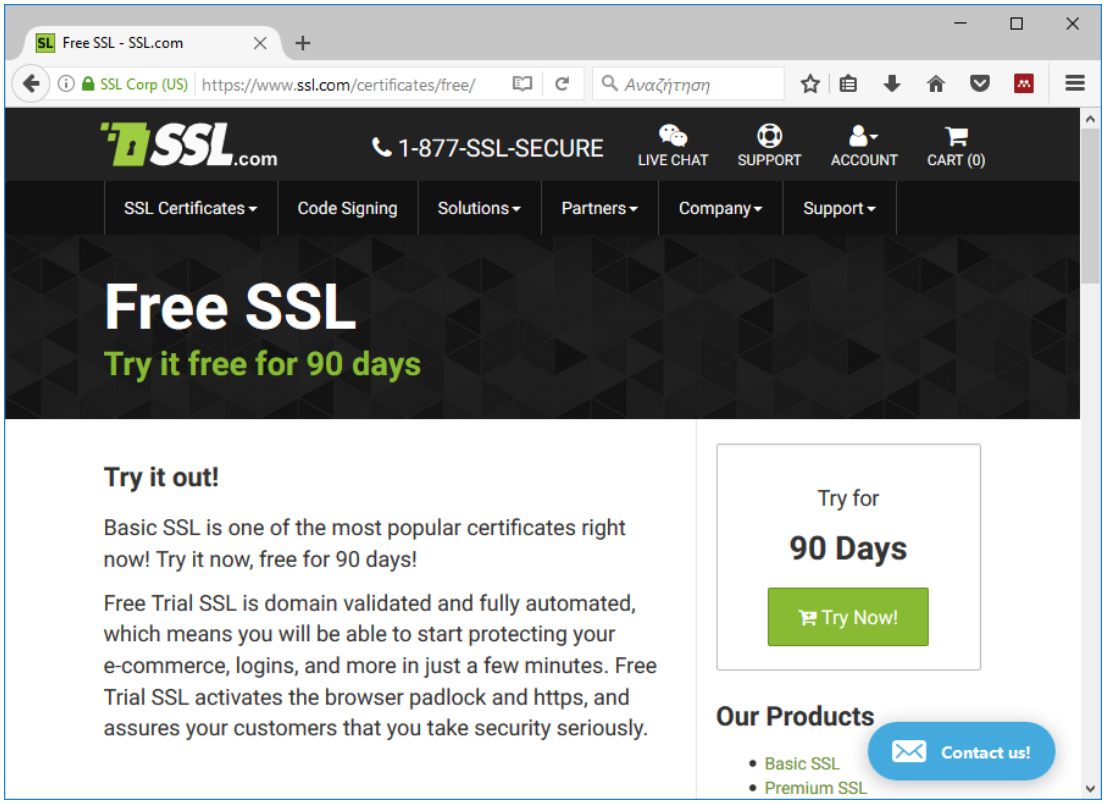


Figure 46. Free SSL Certificate for 90 Days,

SSL certificates allow secure connections to be established between authorized traps and the web server by encrypting all communication. The trap sends via POST, the sensor data values along with a Pre-shared secret hash/key that authorizes it to post data.

3.4.2. Database

The database schema was designed with the free online tool **dbdiffo** database schema designer https://dbdiffo.com/dbdiffo/ and is shown in the following Figure 47. Database schema design.

Figure 47. Database schema design

The same online service provides a means to export the DB schema in SQL format to easily import it into a database server.

Table 1. SQL code that creates the DB Schema in MySQL database server

```
CREATE TABLE `trap` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `Name` varchar(20) NOT NULL,
  `hashID` varchar(32) NOT NULL,
  `location` varchar(30),
  `Latitude` decimal(10,8),
  `longitude` decimal(11,8),
  CONSTRAINT pk_traps PRIMARY KEY (`id`)
);

CREATE TABLE `trap_data` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `trap_id` bigint NOT NULL,
  `datetime_uploaded` timestamp NOT NULL,
  `datetime_data` timestamp NOT NULL,
  `temperature` decimal(4,1),
  `humidity` decimal(5,2),
  `windspeed` smallint,
  `LightLUX` smallint unsigned,
  `LightIR` smallint unsigned,
  `LightBB` smallint unsigned,
  `BarometricPressure` decimal(4,2) unsigned,
  `batteryVoltage` decimal(4,2) unsigned,
  CONSTRAINT pk_trap_data PRIMARY KEY (`id`)
);
```

```
CREATE TABLE `trap_image` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `trap_id` bigint,
  `imagefile` varchar(100),
  `datetime_uploaded` timestamp,
  `datetime_photo` timestamp,
  `insectCountOCV` smallint unsigned,
  `insectCountHuman` smallint unsigned,
  `datetime_checked` datetime,
  CONSTRAINT pk_trap_image PRIMARY KEY (`id`)
);

ALTER TABLE `trap_data` ADD CONSTRAINT
`fk_trap_data_`
  FOREIGN KEY (`trap_id`) REFERENCES `trap` (`id`)
;

ALTER TABLE `trap_image` ADD CONSTRAINT
`fk_trap_image_`
  FOREIGN KEY (`trap_id`) REFERENCES `trap` (`id`)
;
```

3.4.3. Sending data to the HTTP server

The Raspberry Pi executes a cronjob, at fixed intervals and sends data to the webserver. From the web server side, a script is setup that script contains the logic and is responsible for reading all individual sensor values, storing the values in a text file and if possible transmit them to the Web server.

In case there is no internet connection at the time of the data collection all readings are kept in a folder to be transmitted in bulk later on when the trap eventually gets connected to the internet. When data is successfully transmitted it is moved to another directory and kept as an archival copy of all sensor data locally, just in case something goes wrong on the web server side, sort of a backup mechanism.

Provisions have been made to monitor disk capacity in order to avoid losing data due to insufficient disk storage space.

HTTP POST queries containing all parameters we want to transmit and sends them to the remote web server. The domain used is ***thesis.karapatakis.gr*** and for security purposes a Secure Sockets Layer (SSL) certificate, also called digital certificate, was issued.

The HTTP POST is send to the following URL: `"https://thesis.karapatakis.gr/api_data"` using code used

3.4.4. Apache HTTP and web User Interface

To facilitate the best representation of the data collected, the web interface that was designed shows a map of the geographical region where the traps have been placed with the location of each trap placed noted with a red marker symbol.

By clicking on each marker, the region right below the map is updated, showing a graph for each of the parameters send by the trap sensors. Unauthenticated users are shown limited amount of information.

The detailed data already sent to the Web Server is stored in the Database and is presented with graphs of the last 500 values.

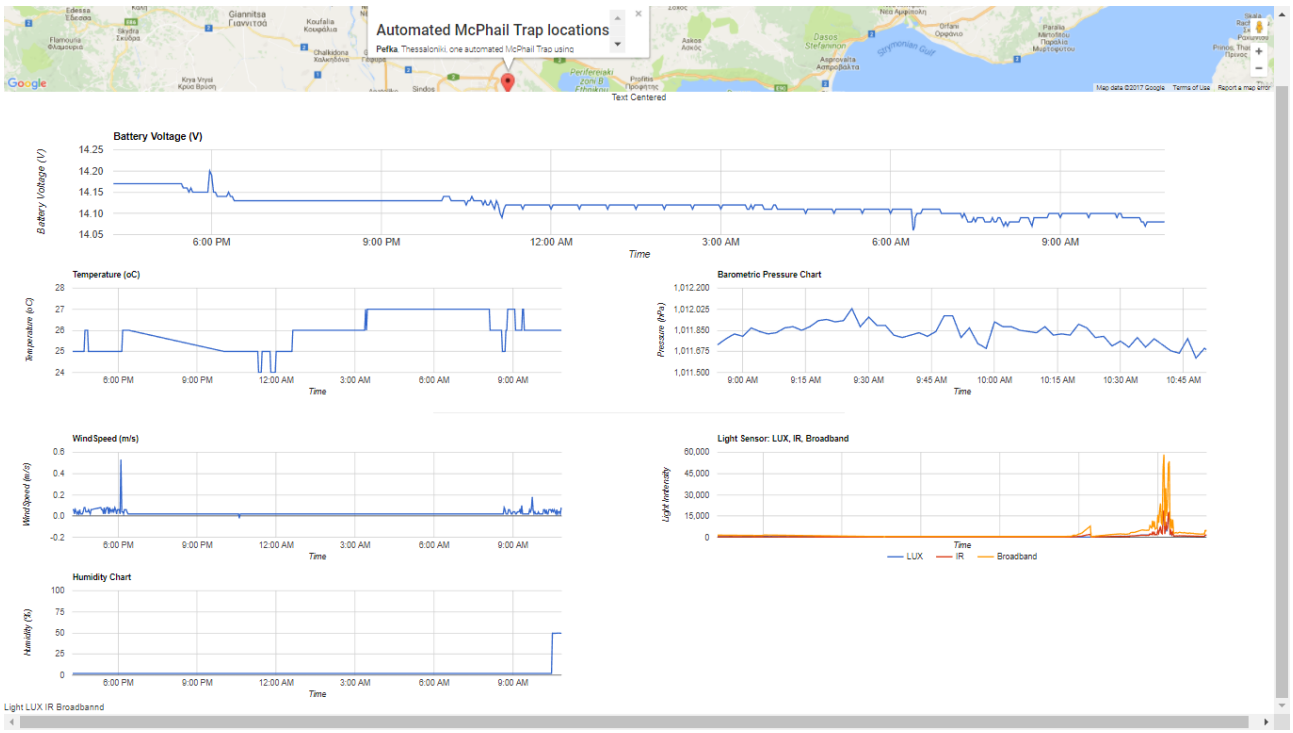
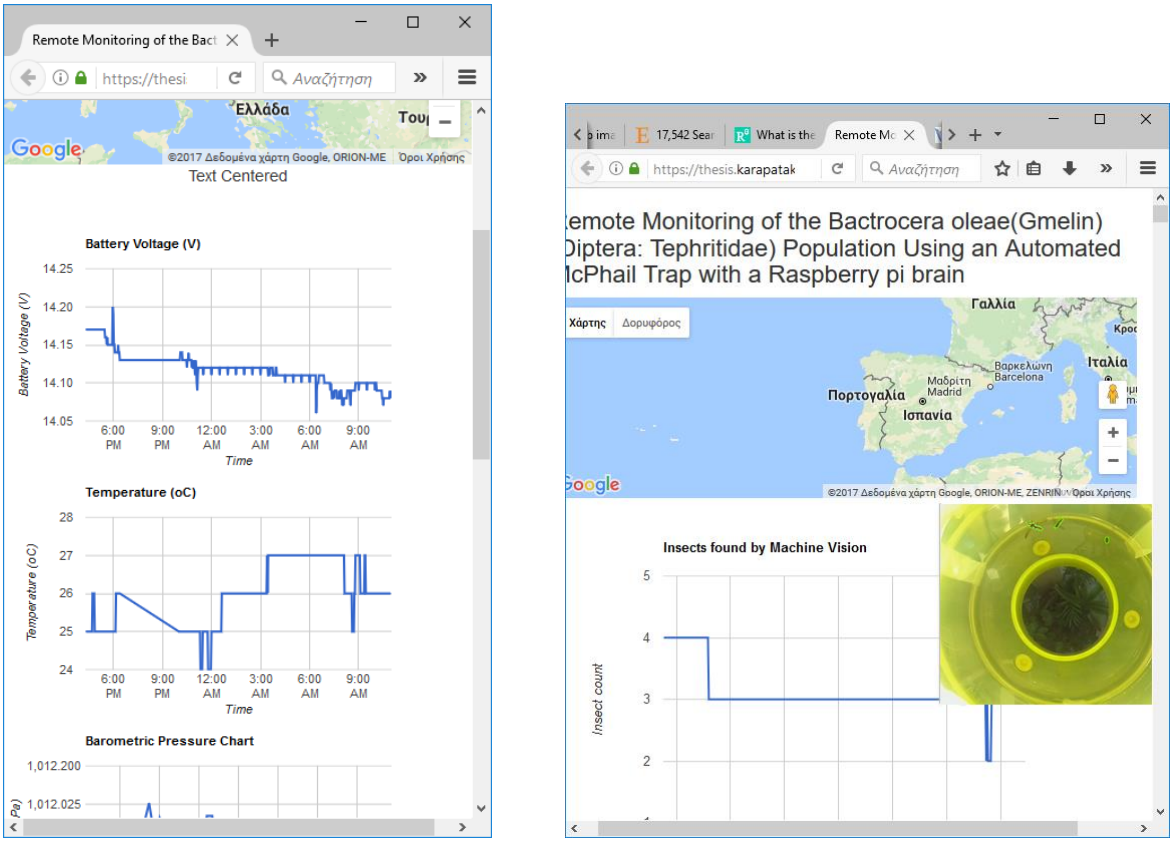


Figure 48. Screenshot of the web server interface



a) b)

Figure 49. a) Responsive design adapts to mobile device's screen, b) graph and images collected from the vision sensor

4. RESULTS

The construction of a McPhail trap using a Raspberry Pi as the main processing unit is feasible and the processing power is adequate to support the machine vision operations that run locally. The system functioned properly for prolonged periods of time, sending both processed images and other sensor data to the webserver at regular intervals, as defined by cronjobs. During the development period, a great number data has been collected that can be used to better finetune parameters of the system.

The system with the current setup and a 12V car battery as a power source, operates without shutting down or going into sleeping mode for over a week (as mentioned there is no support for ACPI low power). One week is an adequate time-period that could co-inside with the time that the trap should be cleaned and the liquid be renewed.

This automatic McPhail trap has a dual task, to record environmental data and capture images. The two tasks have very different sampling demands. Environmental sensors should be queried every couple of minutes, but changes to the number of insects are made with a slower rate and one image captured every hour is sufficient to show population growth.

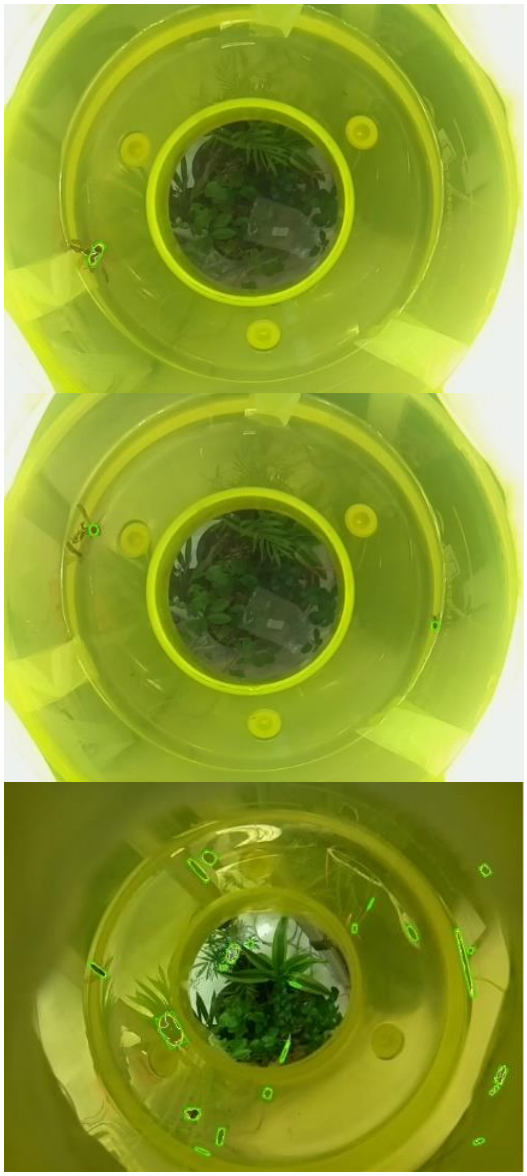
One of the main problems, initially underestimated, was to integrate all the electronics that had different power requirements in respect to voltage and current draw. Especially the GSM modem that was found to be the most demanding, since it needs to be powered with a voltage of 5V and when communications are established with the mobile telephone base station a peak current of 2Amps is required.

machine vision is subject to constant testing and code with improvements are incorporated to increase detection accuracy and overall robustness.

4.1 Image processing problems faced

During the development phase, many problems were faced. Some of the most significant ones are shown below:

a. Falsely detected four (4) insects instead of three because a wasp caught in the trap had stripes that fooled the algorithm.



b. Falsely detected one (1) insect instead of two they were too close and one was not afloat.

c. Falsely detected one (1) insect instead of two because one insect was not afloat and could not be discerned from the background.

d. a) The center hole was not detected, tht led to many dark regions being included in the thresholding process and many false positive detected.

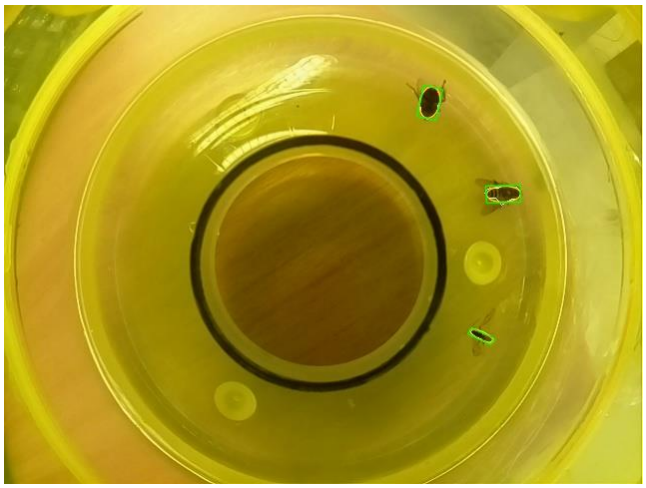
Figure 50. Machine vision problems in insect detection algorithm

Some things worth to mention are

- the longer the insects stay in the liquid, the deeper they are submerged into the solution and that may lead to falsely reporting fewer insects.
- Light intensity greatly affected the robust detection of the center hole. Under some lighting conditions the inner hole was not detected, for example when the sun rays hit the trap and the Hough circles algorithm returned no matches. Hard coding the dimensions in our program would yield better results.
- When wasps or other striped insects are caught in the trap they might be detected as multiple insects.
- Another problem that was faced was to select the proper, thresholding value. This was especially a problem during the development stage until we decided to always turn on the LED lights to aid in the image processing functions.

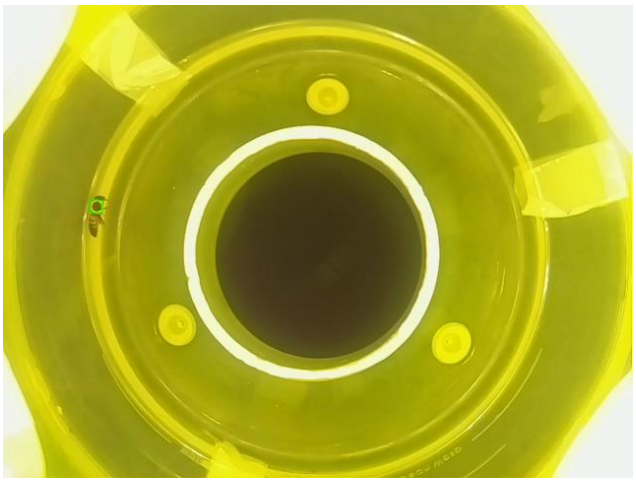
4.2. Image processing good results

The system was tested extensively in laboratory conditions. In the following Figure 51, one can see the output of the machine vision algorithm; the captured images were taken in the controlled environment of our development laboratory.



a. Algorithm detects correct number of insects, when initial tests are conducted in controlled conditions

b. Algorithm correctly detects one insect in trap, test also done in controlled environment.



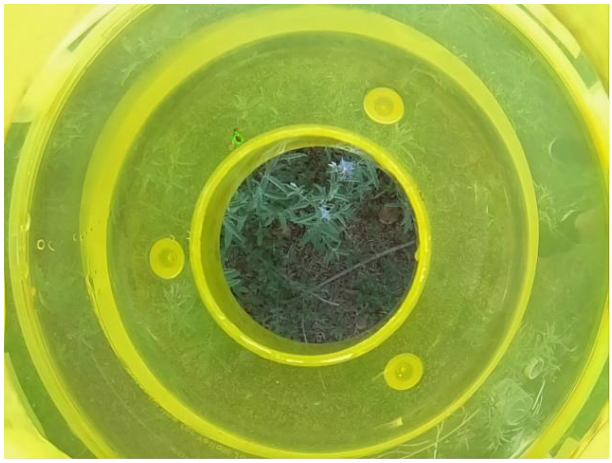
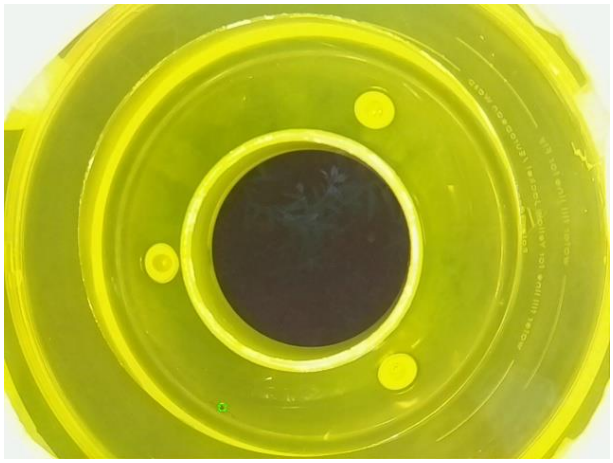
c. Algorithm correctly detects one insect in at night using LED Lights

Figure 51. Machine vision successful attempts in controlled environment

After the successful tests in our laboratory environment, the system was also tested in the field. It was placed under trees and left running for extended periods of time.

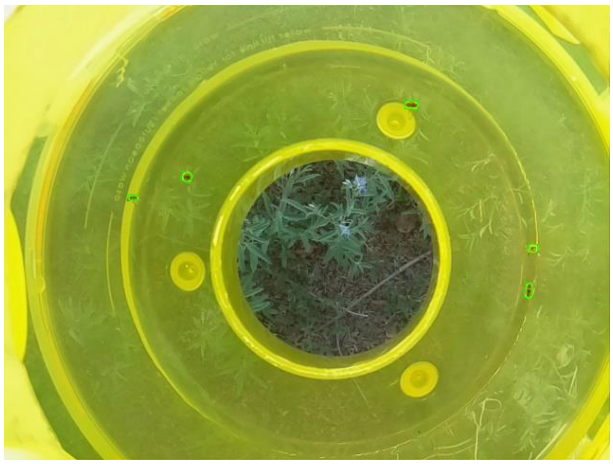
The images in Figure 52, shows the results of the machine vision algorithm with real Bactrocera oleae (Gmelin) (Diptera: Tephritidae) dacus fruit flies in the trap.

captured by the trap camera.

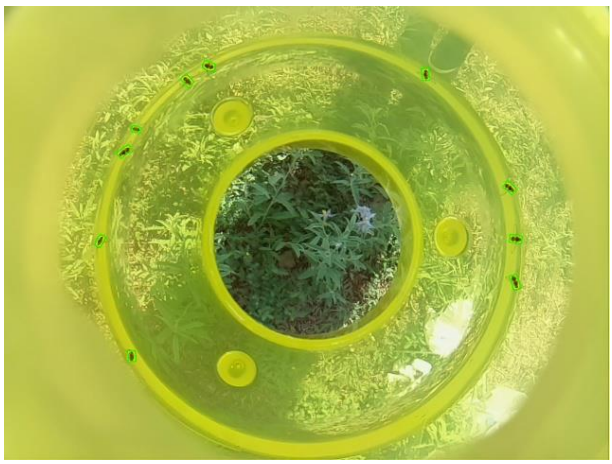



a. Algorithm detects one insect in trap, left image taken during the day and night time with LED lighting.

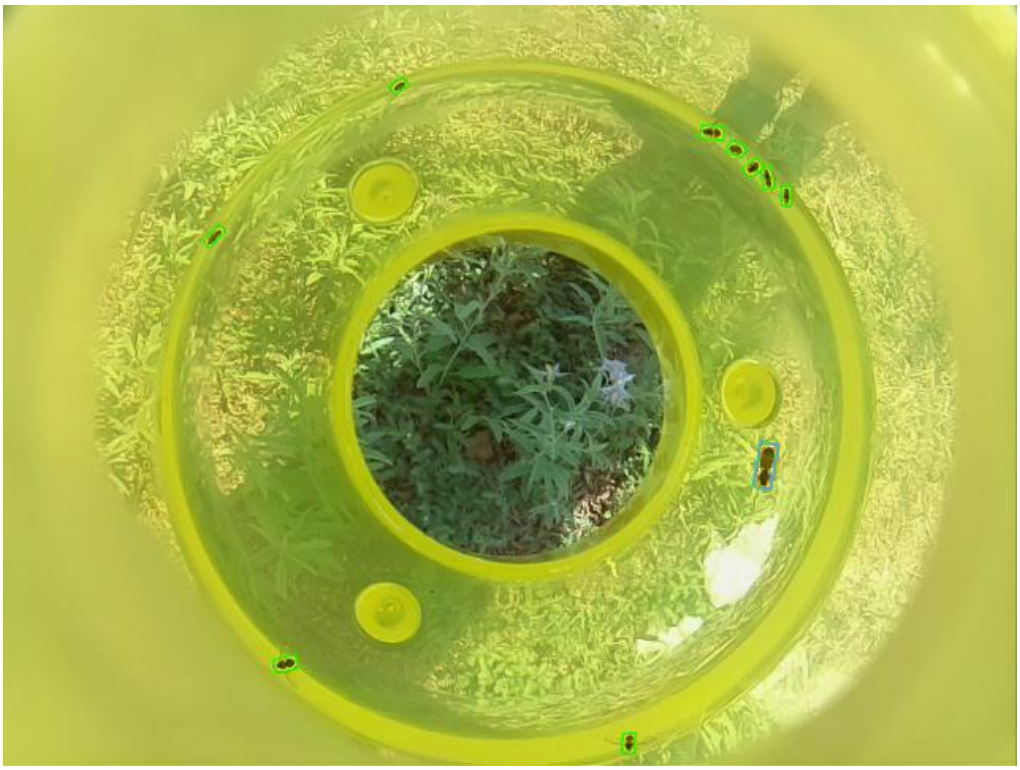

b. Algorithm correctly detects two insects in trap on two different days and time of day.



c. Algorithm detects five insects in the trap



c. Algorithm detects ten insects in the trap



c. Algorithm detects nine flies in the trap and one other insect marked with different color (blue). The insect marked with the blue outline is discarded because its contour covers a larger area than the predefined threshold.

Figure 52. Machine vision algorithm successful dacus fruit fly in the field

5. CONCLUSIONS AND FUTURE WORK-

The success of the whole project heavily relies on the proper operation of the machine vision accuracy and reliable operation of the system, as a whole. Although the trap functions properly, a lot of things could be altered to make it more efficient and allow the trap to work in the harsh weather conditions of an open field. The things that have been identified but were not addressed by this first version of the trap are the following:

- **Weather proof design**. The trap is not water proof, some precautions have been taken to protect some sensitive components but the design and usage of system that would allow the trap to withstand rain is mandatory.
- **CSI Camera connection with the Raspberry Pi**. The camera is currently connected to the Raspberry Pi using a ribbon cable that could be easily be snapped by branches or damaged by UV radiation. An more rugged alternative must be found and in the same time the maximum working distance of that cable must be determined.
- **Battery life.** As far as battery life is concerned:
  - a more efficient switching power supply with voltage regulators should be used.
  - Since the Raspberry Pi does not support low power mode, a timer circuit should be used to turn it on at predefined intervals, and after querrying the sensors it should power down to conserve battery power.
- **Safety**.
  - An I2C Isolator like the Texas Instruments ISO1541, isolation barrier for I2C devices should be used. That would provide more than 2500V of isolation between the two sides of the device and also provides resistance to arcing and breaking the circuit.
  - Since the device is left unattended in the field, some kind of tamper silent alarm should be implemented.
- **Add additional environmental sensors.**
  - Some sensors that could prove to be quite useful and even help producers yield better crops are those related to rainfall and soil moisture. Knowing when to apply water to olive trees is important. Prolonged drought may lead to small crop or even in some cases no produce at all. In a future iteration, the system should be fitted with a rain gauge sensor, that could measure the micro-climate of the region and alert the farmers of the best time to irrigate their crops.
  - Select sensors with better resolution (current temperature sensors used, can only report integer steps).

Camera used. A wide range of vision sensors could be tested to assess whether they could perform better in daytime and low lighting conditions. One of the camera sensors that could be tested is the infrared Camera Module v2 (Pi NoIR).

- **Testing and evaluation in the field.** The standard procedure for the adoption of the proposed system, in the field is the extensive testing and evaluation using standard protocols in the field. Therefore in order to adopt the proposed approach we must test the device using procedures similar to the ones described in (Doitsidis et al., 2016).

BIBLIOGRAPHY

Ada, Lady. (2014). Adding a Real Time Clock to Raspberry Pi, 1–17.

Adafruit ADS1015. (2012). ADS1015 12-Bit ADC - 4 Channel with Programmable Gain Amplifier.

Adafruit Anemometer. (2014). Anemometer Wind Speed Sensor w/Analog Voltage Output.

Adafruit BMP280. (2015). Adafruit BMP280 I2C or SPI Barometric Pressure & Altitude Sensor.

AdafruitAM2315. (2013). AM2315 – Encased I2C Temperature/Humidity Sensor.

Adafruit RTC. (2016). Adafruit PCF8523 Real Time Clock Assembled Breakout Board ID: 3295.

Adafruit TSL2561. (2014). Adafruit TSL2561 Digital Luminosity/Lux/Light Sensor Breakout.

Adafruit Wind Speed Sensor w Analog Voltage output (Product Manual). (2016).

Amvrazi, E. G., & Albanis, T. A. (2009). Pesticide residue assessment in different types of olive oil and preliminary exposure assessment of Greek consumers to the pesticide residues detected. *Food Chemistry*, *113*(1), 253–261. http://doi.org/10.1016/j.foodchem.2008.06.073

Angioni, A., Porcu, L., & Pirisi, F. (2011). LC/DAD/ESI/MS Method for the Determination of Imidacloprid, Thiacloprid, and Spinosad in Olives and Olive Oil after Field Treatment. *Journal of Agricultural and Food Chemistry*, *59*(20), 11359–11366. http://doi.org/10.1021/jf2028363

Bactrocera oleae - bait application. (2012). *EPPO Bulletin*, *42*(3), 431–433. http://doi.org/10.1111/epp.2615

Bechar, I., Moisan, S., Pulsar, E. P. I., & Antipolis-mediterranee, I. S. (2010). On-line counting of pests in a greenhouse using computer vision, 1–4.

Boissard, P., Martin, V., & Moisan, S. (2008). A cognitive vision approach to early pest detection in greenhouse crops. *Computers and Electronics in Agriculture*, *62*(2), 81–93. http://doi.org/10.1016/j.compag.2007.11.009

Bradski, G., & Kaehler, A. (2013). *Learning OpenCV*. *Bradski, Gary Kaehler, Adrian Gary Bradski* (Vol. 53). http://doi.org/10.1017/CBO9781107415324.004

Brahmbhatt, S. (2013). *Practical OpenCV*. *Technology in Action*. Technology in Action. http://doi.org/10.1007/978-1-4302-6080-6_11

Broumas, T. (1994). Olive fruit fly. Review of its biology and its chemical control. *Agrotypos*, (8), 26–31.

Broumas, T. (1995, January). Olive fruit fly. Biological and Biotechnical control methods. *Agrotypos*, 44–54.

Chaubey, A. K. (2016). Comparison of The Local and Global Thresholding Methods in Image Segmentation, (1), 1–4.

Convolutional neural networks cnns. (2016).

Devi, H. K. A. (2006). Thresholding: A Pixel-Level Image Processing Methodology Preprocessing Technique for an OCR System for the Brahmi Script. *Ancient Asia*, (1), 161–165. http://doi.org/http://doi.org/10.5334/aa.06113

DiCola, T. (2014). FONA Tethering to Raspberry Pi or BeagleBone Black.

Doitsidis, L., Fouskitakis, G., Varikou, K., Rigakis, I., Chatzichristofis, S., Sarantopoulos, I., … Birouraki, A. (2016). Remote Monitoring of the Bactrocera oleae(Gmelin) (Diptera: Tephritidae) Population Using an Automated McPhail Trap. *Computers and Electronics in Agriculture*, (under review).

Earl, B. (Adafruid I. (2016). Adafruit 4-Channel ADC Breakouts. Adafruid Industries.

Fouskitakis, G. N., Doitsidis, L. D., Rigakis, H., & Sarantopoulos, I. (2015). Design and Development of Embedded Systems for Precision Agriculture Applications : The Case of Monitoring Dacus Oleae

Population and Bait Sprays.

Gilbert-López, B., García-Reyes, J. F., Fernández-Alba, A. R., & Molina-Díaz, A. (2010). Evaluation of two sample treatment methodologies for large-scale pesticide residue analysis in olive oil by fast liquid chromatography–electrospray mass spectrometry. *Journal of Chromatography A*, *1217*(24), 3736–3747. http://doi.org/10.1016/j.chroma.2010.04.025

Hale, T. B. (2015). C code to read the AOSONG AM2315 I2C Temperature and Humidity Sensor.

Hang, N. (2012). List of MAT types in OpenCV.

Hawkes, N. J., Janes, R. W., Hemingway, J., & Vontas, J. (2005). Detection of resistance-associated point mutations of organophosphate- insensitive acetylcholinesterase in the olive fruit fly, Bactrocera oleae (Gmelin). *Pesticide Biochemistry and Physiology*, *81*(3), 154–163. http://doi.org/10.1016/j.pestbp.2004.11.003

Hong, K. (2016). mat object image matrix.

Hughes, J. (2017). THE RASPBERRY PI UARTS.

Jacobs, D. (2005). Correlation and Convolution.

Johnson, M. W., Wang, X., Nadel, H., Opp, S. B., Lynn-, K., Stewart-leslie, J., & Kent, M. (2011). High temperature affects olive fruit fl y populations in California ' s Central Valley, *65*(March), 29–33.

Kakani, E. G., & Mathiopoulos, K. D. (2008). Organophosphosphate resistance-related mutations in the acetylcholinesterase gene of Tephritidae. *Journal of Applied Entomology*, *132*(9–10), 762–771. http://doi.org/10.1111/j.1439-0418.2008.01373.x

Kernel. (2012).

Koumpouros, Y., Mahaman, B. D., Maliappis, M., Passam, H. C., Sideridis, A. B., & Zorkadis, V. (2004). Image processing for distance diagnosis in pest management. *Computers and Electronics in Agriculture*, *44*(2), 121–131. http://doi.org/10.1016/j.compag.2004.04.004

Lucas, L. (2010). Image Segmentation, 1–19. http://doi.org/10.1002/9780470097434.ch17

Norton, G. W., Rajotte, E. G., & Gapud, V. (1999). Participatory research in integrated pest management: Lessons from the IPM CRSP. *Agriculture and Human Values*, *16*, 431–439. http://doi.org/10.1023/A:1007608019218

Otsu, N. (1979). A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, *9*(1), 62–66. http://doi.org/10.1109/TSMC.1979.4310076

Pavlidi, E., Yusupova, A., Paya, I., Peel, D. A., Martinez-Garcia, E., Mack, A., & Grossman, V. (2013). *Monitoring housing markets for episodes of exuberance: an application of the Phillips et al. (2012, 2013) GSADF test on the Dallas Fed International House Price Database. Federal Reserve Bank of Dallas Globalization and Monetary Policy Institute*.

PkLab.net. (2017). Install OpenCV 3.2 Python/C++ on Raspberry PI.

Potamitis, I., Rigakis, I., & Fysarakis, K. (2014). The electronic McPhail trap. *Sensors (Basel, Switzerland)*, *14*(12), 22285–22299. http://doi.org/10.3390/s141222285

Raspberry_Pi_Foundation. (n.d.). Raspberry pi Downloads - Raspbian.

RealVNC. (2016). RealVNC Knowledge base.

Reignier, R. (2017). Driver for TI's ADS1015: 12-bit Differential or Single-Ended ADC with PGA and Comparator.

Ruiz-Torres, M., & Montiel Bueno, A. (2002). Efectos del dimetoato usado en aplicaciones terrestres y aéreas sobre la entomofauna de olivar en la provincia de Jaén. *Boletín de Sanidad Vegetal - Plagas*, *28*(4), 525–560. Retrieved from

http://www.mapa.es/ministerio/pags/biblioteca/revistas/pdf_plagas%2FBSVP-28-04-525-560.pdf

Ruiz-Torres, R. M., & Muñoz-Cobo, R. J. (1997). Efectos de insecticidas en la entomofauna del olivar. In *VIII Simposium Científico-Técnico del Olivar, EXPOLIVA* (p. 97).

Santos, S. A. P., Pereira, J. A., Torres, L. M., & Nogueira, A. J. A. (2007). Evaluation of the effects, on canopy arthropods, of two agricultural management systems to control pests in olive groves from north-east of Portugal. *Chemosphere*, *67*(1), 131–139. http://doi.org/10.1016/j.chemosphere.2006.09.014

Santos, V. M. R. dos, Donnici, C. L., DaCosta, J. B. N., & Caixeiro, J. M. R. (2007). Compostos organofosforados pentavalentes: histórico, métodos sintéticos de preparação e aplicações como inseticidas e agentes antitumorais. *Química Nova*, *30*(1), 159–170. http://doi.org/10.1590/S0100-40422007000100028

Skouras, P. J., Margaritopoulos, J. T., Seraphides, N. A., Ioannides, I. M., Kakani, E. G., Mathiopoulos, K. D., & Tsitsipis, J. A. (2007). Organophosphate resistance in olive fruit fly,Bactrocera oleae, populations in Greece and Cyprus. *Pest Management Science*, *63*(1), 42–48. http://doi.org/10.1002/ps.1306

Smith, W. S. (2002). *The Scientist and Engineer's Guide to Digital Signal Processing*. Newnes.

Solis-Sánchez, L. O., Castañeda-Miranda, R., García-Escalante, J. J., Torres-Pacheco, I., Guevara-González, R. G., Castañeda-Miranda, C. L., & Alaniz-Lumbreras, P. D. (2011). Scale invariant feature approach for insect monitoring. *Computers and Electronics in Agriculture*, *75*(1), 92–99. http://doi.org/10.1016/j.compag.2010.10.001

Sopwith. (2014). AOSONG AM2315 Temperature Sensor Implementation Guide.

Stack Overflow Community. (2013). debugging c++ programs with gdb.

Szeliski, R. (2010). Computer Vision : Algorithms and Applications.

Thomas. (2016). Automatically update the hwclock with NTP when I have internet connection.

Varikou, K., Alexandrakis, V., & Kalaitzaki, A. (2004). Study of the effectiveness of Mass Trap- ping and new products for the control of the olive fruit fly Bactrocera oleae (Gmelin) (Diptera: Tephritidae) in organic olive groves. In *Proceedings of the Premier Seminaire International sur les Biotechnologies et Qualite des produits de l'Olivier dans le basin Mediterraneen* (p. 90).

Varikou, K., Alexandrakis, V., Mavrotas, K., & Kouletakis, A. (2004). New records for control of Olive fruit fly Bactrocera oleae Gmelin. In *Proceedings of the 5th International Symposium on Olive Growing,* (p. 59).

Varikou, K., Garantonakis, N., & Birouraki, A. (2015). Residual attractiveness of various bait spray solutions to Bactrocera oleae. *Crop Protection*, (68), 60–66.

Varikou, K., Garantonakis, N., Birouraki, A., Ioannou, A., & Kapogia, E. (2016). Improvement of bait sprays for the control of Bactrocera oleae (Diptera: Tephritidae). *Crop Protection*, (81), 1–8.

Varikou, K., N., G., & Birouraki, A. (2014). Comparative field studies of Bactrocera oleae baits in olive orchards in Crete. *Crop Protection*, (65), 238–243.

Weisstein, E. (2012). Gaussian Function Plot.

wikipedia.org. (2017). Beaufort scale.

Yang, X., Shen, X., Long, J., & Chen, H. (2012). An Improved Median-based Otsu Image Thresholding Algorithm. *AASRI Procedia*, *3*, 468–473. http://doi.org/10.1016/j.aasri.2012.11.074

Yao, Q., Lv, J., Liu, Q. jie, Diao, G. qiang, Yang, B. jun, Chen, H. ming, & Tang, J. (2012). An Insect Imaging System to Automate Rice Light-Trap Pest Identification. *Journal of Integrative Agriculture*, *11*(6), 978–985. http://doi.org/10.1016/S2095-3119(12)60089-6

APPENDICES

APPENDIX A: Code and Command Listings

A.1. Updating the Operating System

Listing 3. The first update of the Raspbian operating System

```
# the Linux shell ignores everything that comes after the hash sign #
# so lines beginning with a hash are comments
#
# To be granted system administration privileges, we must become root
# depending on when you last logged in you may be required to type in your
# password again
sudo su –
# Update our local copy of apt package management repositories
apt-get update
# check for (and install without confirmation (-y)) all operating system updates
apt-get -y upgrade
#upgrade the raspberry pi firmware
sudo rpi-update
 *** Raspberry Pi firmware updater by Hexxeh, enhanced by AndrewS and Dom
 *** Performing self-update
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 13403  100 13403    0     0  21019      0 --:--:-- --:--:-- --:--:-- 21040
 *** Relaunching after update
 *** Raspberry Pi firmware updater by Hexxeh, enhanced by AndrewS and Dom
 *** We're running for the first time
 *** Backing up files (this will take a few minutes)
 *** Backing up firmware
 *** Backing up modules 4.9.24-v7+
###############################################################
This update bumps to rpi-4.9.y linux tree
Be aware there could be compatibility issues with some drivers
Discussion here:
https://www.raspberrypi.org/forums/viewtopic.php?f=29&t=167934
###############################################################
 *** Downloading specific firmware revision (this will take a few minutes)
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   168    0   168    0     0    247      0 --:--:-- --:--:-- --:--:--   248
100 53.9M  100 53.9M    0     0   522k      0  0:01:45  0:01:45 --:--:-- 96698
 *** Updating firmware
 *** Updating kernel modules
 *** depmod 4.9.30+
 *** depmod 4.9.30-v7+
 *** Updating VideoCore libraries
 *** Using HardFP libraries
 *** Updating SDK
 *** Running ldconfig
 *** Storing current firmware revision
 *** Deleting downloaded files
 *** Syncing changes to disk
 *** If no errors appeared, your firmware was successfully updated to
684be4bc8cc343f60fdc3240c6d55d41d0a5b56c
 *** A reboot is needed to activate the new firmware
```

A.2 Enabling kernel modules at boot time

Listing 4. Edit the file /etc/modules-load.d/modules.conf

```
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
#
# enable the use of devices on the i2c bus
# (is a symlink to the file /etc/modules)
i2c-dev

# enable the use of the raspberry pi CSI Camera
bcm2835-v4l2
```

The following lines need to be added to the /boot/config.txt file to allow for a greater display resolution when connecting remotely to a headless raspberry pi using RealVNC for example (RealVNC, 2016).

Listing 5. Edit the file /boot/config.txt

```
# Tells your Pi an HDMI display is attached.
hdmi_force_hotplug=1
hdmi_ignore_edid=0xa5000080
hdmi_group=2
# Forces a resolution of 1024x768 at 60Hz. See here for more options.
hdmi_mode=16
```

A.3 Install some command line utilities and customizations

Listing 6. a) Execute the commands and b) add the following instructions to ~/.vimrc

```
# install the most commonly used utilities
#
apt-get -y install vim wget rsync nfs-utils nfs-utils-lib bind-utils

# make editing easier in vim (color scheme etc)
# edit file ~/.vimrc
vim ~/.vimrc

# add the following instructions
syntax on
set number
set expandtab
set tabstop=4
```

```
set backup
colorscheme elflord
```

A.4 Sending email report when unit has internet connectivity

To send an automated email every time the system connects to the internet, a scheduled job (cron job) was created, that runs the script `/root/sendemail.py`

Listing 7. Edit the cronjob entries for user root by running root@trap$ *crontab -e*

```
# at the end of the file add the following two entries
#
# on reboot disable the wireless interface (wlan0)
@reboot sudo ifdown wlan0
#
# every minute run the IP connectivity.py python script that determines
# if there have been any changes in internet connectivity since the last time
# it was executed (compares information stored in a temporary file)
* * * * * /root/ipconnectivity.py

```

Listing 8. Edit the python script file /root/sendemail.py that senses internet connectivity changes

```
#!/usr/bin/python

import subprocess
import smtplib
from datetime import datetime
import subprocess
import socket
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from requests import get



def uptime():
    raw = subprocess.check_output('uptime').replace(',','')
    days    = 0
    hours   = 0
    minutes = 0
    if "days" in raw:
        days = int(raw.split()[2])
        hours, minutes = map(int,raw.split()[4].split(':'))
    elif 'min' in raw:
```

```
        hours = 0
        minutes = int(raw[4])
    else:
        hours, minutes = map(int,raw.split()[2].split(':'))
    totalsecs = days*24*60*60 + hours*60*60 + minutes*60
    return totalsecs

my_uptime=uptime()
#print 'System uptime of %d seconds : ~ %s minutes' %
(my_uptime,int(my_uptime)/60)



today = datetime.now()
print '%s' % today.strftime('%x %X')

ip = subprocess.check_output(['hostname', '-I'])
publicIP      = 'No Network connection :('
try:
        publicIP     = get('https://api.ipify.org').text
except :
        print 'Connection aborted.  gaierror(-2, Name or service not known'


publicIPLast = "You have to send Email if nothing updates this!!"
try:
        publicIPLast = open('/tmp/current_public_ipaddress', 'r').read()
except IOError:
        print "First Run, no ip address file."

if publicIP == publicIPLast:
        print "Same IP exiting, without sending anything..."
        print "Private: %s" %       ip[:-2].decode("UTF-8")
        print "Public : %s" % publicIP[:].decode("UTF-8")
        quit()
else:
        file = open("/tmp/current_public_ipaddress","w")
        file.write(publicIP)
        file.close()

to = 'Fistemail@address.here'
cc = 'second_email@yahoo.gr'
gmail_user = 'username@gmail.com'
gmail_password = 'password_'

smtpserver = smtplib.SMTP('smtp.gmail.com', 587)
smtpserver.ehlo()
smtpserver.starttls()
smtpserver.ehlo()
smtpserver.login(gmail_user, gmail_password)

BODY = '<html><body><h2>Raspbery Pi IP Changed</h2>\n'
BODY = BODY+'<pre>Date......: %s\n' % today.strftime('%b %d %Y')
BODY = BODY+'Time......: %s\n'  % today.strftime('%X')
BODY = BODY+'Private IP: %s\n'  % ip[:-2].decode("UTF-8")
BODY = BODY+'Public  IP: %s\n'  % publicIP[:].decode("UTF-8")
BODY = BODY+'System uptime: (hr)  %.1f \n' % float(my_uptime/60/60)
BODY = BODY+'               (min) %d \n' % int(my_uptime/60)
BODY = BODY+'               (sec) %d \n' % int(my_uptime)
BODY = BODY+'</pre>\n&copy; 2017 ikarapatakis@yahoo.gr</body></html>'
```

```
83
84
85  msg = MIMEMultipart('alternative')
86
87  HTML_BODY = MIMEText(BODY,'html')
88
89  msg.attach(HTML_BODY)
90  msg['Subject'] = 'RPi Trap IP: %s  Public IP: %s' % (ip[:].decode("UTF-
    8"),publicIP[:].decode("UTF-8"))
91  msg['From'] = gmail_user
92  msg['To'] = to
93  msg['Cc'] = cc
94  smtpserver.sendmail(gmail_user, [to], msg.as_string())
95  smtpserver.quit()
96
97  # script adapted to send HTML rich email from
98  # https://www.spritecloud.com/2010/03/creating-and-sending-html-e-mails-with-python/
99  # https://gist.github.com/carlkibler/1019870
100
```

A.5 Enable the I2C Real Time Clock module

To enable the I2C Real Time Clock module we must declare it in the device tree overlay so that it is enabled on boot; to do that we must edit the **/boot/config.txt** file and add the following lines at the end:

Listing 9. Edit the configuration file: *vim /boot/config.txt*

```
1  # at the end of the file add the following entry
2  #
3  # on boot enable the RTC module
4  dtoverlay=i2c-rtc,pcf8523
```

After rebooting the system and if the module seems to function properly, we must disable the fake hardware clock installed by the OS by running the following commands (Ada, 2014):

Listing 10.Remove fake hardware clock support from the Raspbian OS

```
1  sudo apt-get -y remove fake-hwclock
2  sudo update-rc.d -f fake-hwclock remove
```

one last step is to comment out some lines on the **/lib/udev/hwclock-set** file:

Listing 11.comment out three lines by editing the file: *vim /lib/udev/hwclock-set*

```
1  #if [ -e /run/systemd/system ] ; then
2  # exit 0
3  #fi
```

After rebooting the Raspberry, and checking the i2c devices it should show UU under the I2C address, that means that it is a reserved address and it's being used.

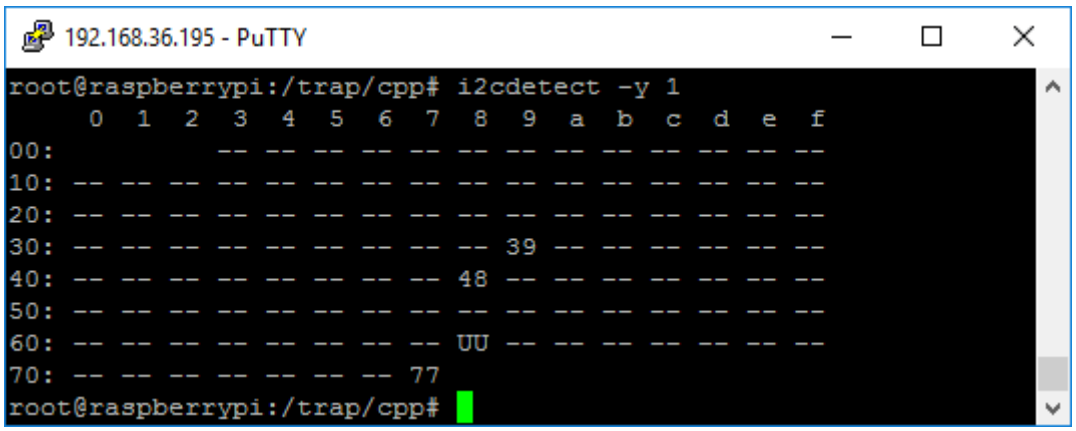


Figure 53. The RTC is now used by the system and that is reflected by the **UU** symbol in place of the RTC address #**68**

A.6 Automatically update the HW-clock with NTP Internet server

All Real time clocks drift due to impurities in their crystal oscillator, therefore we can create a script that checks the time stored in the RTC against a Network Time Protocol server when we connect to the internet (Thomas, 2016).

Listing 12.bash shell script that reads the date and time from an NTP server and saves it in our hardware clock (RTC module)

```
1   #This bash script gets the time stored in the hardware clock, and the time from
    and NTP Server, verifying that they are the same.
2   #!/bin/bash
3
4   # Location of logfile
5   LOGFILE="/home/pi/hwcsync/ntplog.log"
6
7   if [ ! -f $LOGFILE ]; then
8     touch $LOGFILE
9   fi
10
11  # Set the maximum allowed difference in seconds between Hw-Clock and Sys-Clock
12  maxDiffSec="2"
13
14  msgNoConnection="No connection to time-server"
15  msgConnection="Connection to time-server"
16
17  # Check for NTP connection
18  if ( ntpq -p | grep -q "^*"  ); then
19    echo $msgConnection
20    echo "---------------------------------"
21
22    secHwClock=$(sudo hwclock --debug | grep "^Hw clock time" | awk '{print $(NF-
    3)}')
23    echo "HwClock: $secHwClock sec"
24
25    secSysClock=$(date +"%s")
26    echo "SysClock: $secSysClock sec"
27    echo "---------------------------------"
28
29    secDiff=$(($secHwClock-$secSysClock))
39
```

```
  # Compute absolute value
  if ( echo $secDiff | grep -q "-" ); then
     secDiff=$(echo $secDiff | cut -d "-" -f 2)
  fi

  echo "Difference: $secDiff sec"

  msgDiff="HwClock difference: $secDiff sec"
  if [ "$secDiff" -gt "$maxDiffSec" ] ; then
    echo "--------------------------------"
    echo "The difference between Hw- and Sys-Clock is more than $maxDiffSec sec."
    echo "Hw-Clock will be updated"

    # Update hwclock from system clock
    sudo hwclock -w
    msgDiff="$msgDiff --> HW-Clock updated."
  fi
  if !(awk '/./{line=$0} END{print line}' $LOGFILE | grep -q "$msgConnection") ||
[ "$secDiff" -gt "$maxDiffSec" ]; then
    echo $(date)": "$msgConnection". "$msgDiff >> $LOGFILE
  fi
else
  # No NTP connection
  echo $msgNoConnection
  if !(awk '/./{line=$0} END{print line}' $LOGFILE | grep -q "$msgNoConnection");
then
    echo $(date)": $msgNoConnection" >> $LOGFILE
  fi
fi
```

If we add the above bash shell script in our root crontab we can always keep a good date and time, never being of sync for more than two seconds off.

A.7 Read Humidity and Temperature values from the AOSONG AM2315 sensor

This sensor seems to have some problems with the Rpi I2C bus, i2cdetect -y 1, its address is @5c, but it seems unstable, it does not seem to find it all the time.

Therefore we adapted the C code that is used to read the sensor to try to read the sensor and if successful then exit with the readings (Hale, 2015)

Listing 13. C code to read the AOSONG AM2315 I2C Temperature and Humidity Sensor

```
/*
 example code to test am2315 (temp/humid sensor) on i2c bus ted.b.hale@gmail.com
*/
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdint.h>
#include <wiringPi.h>
#include <wiringPiI2C.h>

int main(int argc, char *argv[])
```

```
14 {
15  int n, fd;
16
17   // read request - 3 is the read register command
18   // 0 is the address to start at
19   // 4 is the number of bytes to read
20  unsigned char read_request[3] = {3, 0, 4};
21
22  // buffer for the response: command byte, length byte, 4 bytes data, 2 bytes of
23 checksum
24  unsigned char response[8];
25
26  // dummy data sent to wake up device
27  unsigned char dummy[1] = {0};
28
29  // the final results
39  float humidity, celsius;
31
32  // open the am2315 device using WiringPi
33  // 0x5C is bus address of am2315
34  // fd is the "file descriptor" used in later read and writes
35  fd = wiringPiI2CSetup(0x5c);
36  if (fd==-1)
37  {
38   printf("wiringPiI2CSetup failed\n");
39   return 0;
40  }
41
42  while (1)
43  {
44   // send some data to wake it up
45   n = write(fd, dummy, 1);
46   n = write(fd, dummy, 1);
47
48   // send the read request
49   n = write(fd, read_request, 3);
50   printf("write returned %d bytes\n",n);
51
52   // very short delay to allow device to do data conversion
53   delay(2);
54
55   // read the reaponse
56   n = read(fd, response, 8);
57   printf("read returned %d bytes\n",n);
58
59   // sanity check on data returned
60   // first byte should echo the read requst byte (3)
61   // second byte should indicate 4 bytes of data returned
62   // I don't bother verifying the checksum
63   if ((response[0]!=3) || (response[1]!=4))
64   {
65    printf("i2c response invalid\n");
66    for (n=0; n<8; n++)
67     printf("%02x ",response[n]);
68   }
69   else
70   {
71    // (high byte * 256) + low byte
72    // divide by 10
73    humidity = (256*response[2] + response[3])/10.0;
```

```
74
75    // same as above but mask out the sign bit on the high byte
76    celsius = (256 * (response[4] & 0x7F) + response[5]) / 10.0;
77    // make result negative if the sign bit is set
78    if ((response[4]&0x80)!=0)
79     celsius *= -1.0;
80
81    printf("   humidity = %5.1f%%\n",humidity);
82    printf("temperature = %5.1f\n",celsius);
83   }
84   printf("\n\n\n");
85
86   // wait two second and loop again
87   delay(2000);
88  }
89
90  return 0 ;
91  }
```

And to compile the C code on the Raspberry Pi we must run the following command

Listing 14. Compile C code and link the wiringPi library that is used.

```
1  gcc -lwiringPi -o read read.c
```

A.8 Machine Vision C++ code that detects insects in on the trap's surface

Listing 15. C code to capture an image from the camera, and detect insects

```
1  #include <time.h>
2  #include <stdlib.h>     // for using the function sleep
3  #include <stdio.h>
4  #include <opencv2/opencv.hpp>
5  #include <opencv2/highgui.hpp>
6  #include <opencv2/imgproc/imgproc.hpp>
7  #include <cmath>
8
9  // added after segmentation faults
10 // https://stackoverflow.com/questions/315948/c-catching-all-exceptions
11 #include <exception>
12 #include <typeinfo>
13 #include <stdexcept>
14 #include <getopt.h>  // adapted example from https://codeyarns.com/2015/01/30/how-
   to-parse-program-options-in-c-using-getopt_long/
15
16 using namespace cv;
17 using namespace std;
18
19 std::string get_date(void);
20 std::string get_year_moth_path(void);
21
22 // Declare Variables
23 int lux = -1;
24 std::string  timestamp = "";
25 std::string  id        = "ID001";
26 std::string  path      = "grabbed/"+get_year_moth_path();
27 bool is_verbose        = false;
```

```
28
29
39
31 void PrintHelp() {
32    std::cout << "\n" <<
33    "--lux <n>:              Set number of lux (light intensity)\n"
34    "--timestamp <datetime>: File to write to\n"
35    "--path <fullpath>:      Directory under /trap/ [default=grabbed]\n"
36    "--ID <IDstring>:        Trap ID unique identifier [default=ID001] \n"
37    "--verbose:              Show debug messages / verbose output\n";
38    "--help:                 Show help\n\n";
39    exit(1);
40 }
41
42 void ProcessArgs(int argc, char** argv)
43 {
44     const char* const short_opts = "l:t:p:i:v:h";
45     const option long_opts[] = {
46        {"lux",       1, nullptr, 'n'},
47        {"verbose",   0, nullptr, 'v'},
48        {"timestamp", 1, nullptr, 't'},
49        {"id",        1, nullptr, 'i'},
50        {"path",      1, nullptr, 'p'},
51        {"help",      0, nullptr, 'h'},
52        {nullptr,     0, nullptr,   0}
53     };
54
55     while (true)
56     {
57          const auto opt = getopt_long(argc, argv, short_opts, long_opts, nullptr);
58
59          if (-1 == opt)
60              break;
61
62          switch (opt)
63          {
64          case 'l':
65              lux = std::stoi(optarg);
66              if (is_verbose) std::cout << "LUX set to: " << lux << std::endl;
67              break;
68
69          case 'v':
70              is_verbose = true;
71              if (is_verbose) std::cout << "Verbose is set to true\n";
72              break;
73
74          case 'i':
75              id = std::string(optarg);
76              if (is_verbose) std::cout << "id set to: " << id << std::endl;
77              break;
78
79          case 't':
80              timestamp = std::string(optarg);
81              if (is_verbose) std::cout << "timestamp set to: " << timestamp <<
    std::endl;
82              break;
83
84          case 'p':
85              path = std::string(optarg);
86              if (is_verbose) std::cout << "path set to: " << path << std::endl;
```

```
87              break;
88
89          case 'h': // -h or --help
90          case '?': // Unrecognized option
91          default:
92              PrintHelp();
93              break;
94          }
95      }
96  }
97
98
99  int main(int argc,char ** argv)
100 {
101     timestamp = get_date();
102     if (is_verbose) {
103         cout <<endl << "Lux:       " << lux;
104         cout <<endl << "timestamp: " << timestamp <<endl;
105     }
106     ProcessArgs(argc, argv);
107     if (is_verbose) {
108         cout <<endl << "Lux:       " << lux ;
109         cout <<endl << "timestamp: " << timestamp <<endl;
110     }
111
112     VideoCapture cap(0);
113     if (!cap.isOpened()) {
114        cerr << "ERROR: Unable to open the camera" << endl;
115        return 0;
116     }
117
118     Mat frame;
119     if (is_verbose) cout << "Start grabbing" << endl;
120     cap >> frame;
121     cap >> frame;
122     if (frame.empty()) {
123        cerr << "ERROR: Unable to grab from the camera" << endl;
124        exit;
125     }
126
127     cv::Mat originalHole;
128     cv::Mat gray_image;
129     cv::Mat grayHole;
139     cv::Mat thresh;
131     cv::Mat im_bw;
132
133     frame.copyTo(originalHole);
134     cvtColor( frame, gray_image, CV_BGR2GRAY );
135     gray_image.copyTo(grayHole);
136
137     imwrite( "/trap/"+path+"/"+id+"_"+timestamp+"_raw________.jpg", frame );
138     imwrite( "/trap/"+path+"/"+id+"_"+timestamp+"_gray_______.jpg", gray_image );
139
140     cv::Mat histogramEqualized; cv::equalizeHist(gray_image,histogramEqualized);
141     imwrite( "/trap/"+path+"/"+id+"_"+timestamp+"_histEQ_____.jpg",
    histogramEqualized);
142     cv::Mat blured; GaussianBlur(histogramEqualized, blured, Size(9,9),2,2);
143     imwrite( "/trap/"+path+"/"+id+"_"+timestamp+"_histEQ_blur.jpg",
    histogramEqualized);
144
```

```
    cv::Mat img_masked;
    cv::Mat img_maskedg;
    // https://stackoverflow.com/questions/30065953/mask-color-image-in-opencv-c
    // Create a black colored image, with same size and type of the input color image // bw - gray CV_8U
    // cv::Mat liquidMask = gray_image >255;  //ALL Black. All pixels in the matrix greater than 128 turn white
    cv::Mat liquidMaskg(gray_image.size(), gray_image.type());
    liquidMaskg.setTo(cv::Scalar(0,0,0));
// creates black

    cv::Mat liquidMask(originalHole.size(), originalHole.type());
    liquidMask.setTo(cv::Scalar(0,0,0));
// creates black

    // detect and remove centerhole,
    vector<Vec3f> circles;
    vector<Vec3f> circles2;
    if (is_verbose) cout << "Will try to find Circles, in order to detect the center hole"<<endl;
    //HoughCircles( blured, circles, CV_HOUGH_GRADIENT, 1, gray_image.rows/6, 81, 173, 90 , 130 ); // 90 min radius, 130 max radius.
    HoughCircles( blured, circles, CV_HOUGH_GRADIENT, 1, gray_image.rows/8, 200, 100,90 , 130 ); // 90 min radius, 130 max radius.

  try {
  // Draw the circles detected
     for( size_t i = 0; i < circles.size(); i++ ) {
       Point center(cvRound(circles[i][0]), cvRound(circles[i][1]));
       int radius = cvRound(circles[i][2]);

       circle(gray_image, center, 3, Scalar(0,255,0), -1, 8, 0);// circle center
       circle(gray_image, center, 3, Scalar(0,255,0), -1, 8, 0);// circle center
       circle(originalHole, center, radius+2, Scalar(255,255,255), -1, 8, 0);// circle outline
       //create mask (for color image)
       circle(liquidMask,center , 10+gray_image.rows/2,Scalar(255,255,255),-1, 8, 0);
       circle(liquidMask,center , radius+2, Scalar(0,0,0),-1, 1, 0);
       imwrite( "/trap/"+path+"/"+id+"_"+timestamp+"_liquidMask_.jpg", liquidMask );

       //create mask for gray image
       circle(liquidMaskg,center , 10+gray_image.rows/2,Scalar(255,255,255),-1, 8, 0);
       circle(liquidMaskg,center , radius, Scalar(0,0,0),-1, 1, 0);
       imwrite( "/trap/"+path+"/"+id+"_"+timestamp+"_liquidMaskg_.jpg", liquidMask );

       originalHole.copyTo(img_masked,liquidMask);
       imwrite( "/trap/"+path+"/"+id+"_"+timestamp+"__OriginalMasked_.jpg", img_masked );
       gray_image.copyTo(img_maskedg,liquidMaskg);
       imwrite( "/trap/"+path+"/"+id+"_"+timestamp+"__grayMasked_.jpg", img_maskedg );


       if (is_verbose) cout << "Circle1:"<<endl<<"center : " << center << "\nradius : " << radius ;
     }
```

```
204  }
205  catch (...) {
206        std::exception_ptr p = std::current_exception();
207        std::clog <<(p ? p.__cxa_exception_type()->name() : "null") << std::endl;
208  }
209
210
211  int holeCenterX =0;
212  int holeCenterY =0;
213  if (circles.size() ==1) {
214     holeCenterX = circles[1][0];
215     holeCenterY = circles[1][1];
216  }
217
218  cvtColor(originalHole, grayHole, CV_BGR2GRAY );
219
220  cv::Mat blured2; GaussianBlur(grayHole, blured2, Size(9,9),2,2);
221
222  imwrite( "/trap/"+path+"/"+id+"_"+timestamp+"_grayCircle_.jpg", grayHole );
223  imwrite( "/trap/"+path+"/"+id+"_"+timestamp+"_orinalHole_.jpg", originalHole);
224
225  // try to make only darker pixels visible, Darker pixels have intensity values
226from 0 up to a certain threshhold we define.
227  // Our chosen threshold is going to be 128. So turn anything brighter that 128
228into pure white.
229
230  cv::Mat erase_grayishAreas = grayHole;
231  if (is_verbose) {
232    cout << "Rows: "<< grayHole.rows<< "Cols: "<< grayHole.cols<<endl;
233    cout << "Rows: "<< erase_grayishAreas.rows<< "Cols: "<<
234erase_grayishAreas.cols<<endl;
235  }
236  int threshold_lux_norm_test=100;
237
238  for(int y=0;y<grayHole.rows;y++) {
239    //cout <<endl<<" COL :" << y ; //<<endl;
240    for(int x=0;x<grayHole.cols;x++) {
241        // if current pixel is more than threshold value === 128 set to 255
242        // cout<< "(" << x << ","<<y<<") = "<< (int)grayHole.at<uchar>(x,y);
243        if ((int)grayHole.at<uchar>(y,x)>threshold_lux_norm_test) {
244            erase_grayishAreas.at<uchar>(y,x)=255;
245            //cout << "W";
246        }
247        //cout << endl;
248    }
249  }
250
251 if (is_verbose) cout << "Lets Save it now!"<<endl;
252 //Save custom thresholded image
253 imwrite( "/trap/"+path+"/"+id+"_"+timestamp+"_noblur_eraseGray_.jpg",
254erase_grayishAreas );
255
256
257  GaussianBlur(grayHole, blured, Size(9,9),2,2);
258  threshold(blured, im_bw, 0,255,CV_THRESH_BINARY | CV_THRESH_OTSU);
259  imwrite( "/trap/"+path+"/"+id+"_"+timestamp+"_blur_9x9_gray_.jpg", grayHole );
260  imwrite( "/trap/"+path+"/"+id+"_"+timestamp+"_blur_9x9_thrOtsu__.jpg", im_bw );
261
262     vector<vector<Point> > contours;
263     findContours(im_bw, contours, RETR_LIST, CHAIN_APPROX_NONE);
```

```
264
265
266     /// Get the moments
267     vector<Moments> mu(contours.size() );
268     for( int i = 0; i < contours.size(); i++ )  { mu[i] = moments( contours[i],
    false ); }
269
270     ///  Get the mass centers:
271     vector<Point2f> mc( contours.size() );
272     for( int i = 0; i < contours.size(); i++ )  { mc[i] = Point2f(
    mu[i].m10/mu[i].m00 , mu[i].m01/mu[i].m00 ); }
273
274
275     Mat cimage = Mat::zeros(im_bw.size(), CV_8UC3);
276     Mat cimageOriginal; frame.copyTo(cimageOriginal);
277     int insectsFound=0;
278     for(size_t i = 0; i < contours.size(); i++)
279     {
280          size_t count = contours[i].size();
281          if (is_verbose) cout << " Contour " << i << "  Area: " <<
    contourArea(contours[i]) << " Center : (" << cvRound(mc[i].x) << ","<<
    cvRound(mc[i].y) << ")"<< std::endl;
282
283          // determine if area of contour is too big... it is probably a shadow.
284          if (contourArea(contours[i]) > 400 ) {
285             if  (is_verbose) {
286                    cout << "Skipping Too big " <<endl;
287             }
288             continue;
289          }
290
291           if( count < 20 )
292             continue;
293          Mat pointsf;
294          Mat(contours[i]).convertTo(pointsf, CV_32F);
295          RotatedRect box = fitEllipse(pointsf);
296          if( MAX(box.size.width, box.size.height) > MIN(box.size.width,
    box.size.height)*30 )
297              continue;
298          drawContours(cimage, contours, (int)i, Scalar::all(255), 1, 8);
299          drawContours(cimageOriginal, contours, (int)i, Scalar::all(255), 1, 8);
300          ellipse(cimage, box, Scalar(0,0,255), 1, LINE_AA);
301          ellipse(cimageOriginal, box, Scalar(0,0,255), 1, LINE_AA);
302          ellipse(cimage, box.center, box.size*0.5f, box.angle, 0, 360,
    Scalar(0,255,255), 1, LINE_AA);
303          ellipse(cimageOriginal, box.center, box.size*0.5f, box.angle, 0, 360,
    Scalar(0,255,255), 1, LINE_AA);
304          Point2f vtx[4];
305          box.points(vtx);
306         insectsFound++;
307          for( int j = 0; j < 4; j++ ) {
308              line(cimage, vtx[j], vtx[(j+1)%4], Scalar(0,255,0), 1, LINE_AA);
309              line(cimageOriginal, vtx[j], vtx[(j+1)%4], Scalar(0,255,0), 1,
    LINE_AA);
310          }
311     }
312
313
314
315
```

```
316     //imshow("result", cimage);
317     imwrite( "/trap/"+path+"/"+id+"_"+timestamp+"_blur_11xOtsu_elipses_.jpg",
318cimage );
319     imwrite( "/trap/"+path+"/"+id+"_"+timestamp+"_blur_11xOtsu_elipses.jpg",
320cimageOriginal );
321
322
323
324  cout << "Found:"<< insectsFound << ", image:"<<
325"/trap/"+path+"/"+id+"_"+timestamp+"_blur_11xOtsu_elipses.jpg"  << endl;
326  if (is_verbose) cout << "Closing the camera" << endl;
327  cap.release();
328  destroyAllWindows();
329  if (is_verbose) cout << "bye!" <<endl;
339  return 0;
331}
332
333
334
335std::string get_year_moth_path(void)
336 {
337    time_t now;
338    char the_date[22];
339
340    the_date[0] = '\0';
341
342    now = time(NULL);
343
344    if (now != -1)
345    {
346       strftime(the_date, 22 , "%Y/%m", gmtime(&now));
347    }
348    return std::string(the_date);
349 }
350
351std::string get_date(void)
352{
353   time_t now;
354   char the_date[22];
355
356   the_date[0] = '\0';
357
358   now = time(NULL);
359
360   if (now != -1)
361   {
362      strftime(the_date, 22 , "%Y_%m_%d__%H-%M-%S", gmtime(&now));
363   }
364   return std::string(the_date);
365}
366
```

And to compile the C code on the Raspberry Pi we must run the following command

ffListing 16. Compile C++ code and link the executable handling computer vision functions.

```
1  g++ -std=c++11 $(pkg-config --libs --cflags opencv) -o \
2       grab_removeHole_histEq_OtsuThresh grab_removeHole_histEq_OtsuThresh.cpp
```

A.7 Python scripts that collect sensor values and send them to the webserver

This paragraph describes, the processes mentioned in section Error! Reference source not found.**: 3.2. Operation of the** Trap. The two scripts were created to facilitate the collection of the data from the sensors and are run using cronjobs that are setup under the Linux user "pi".

Listing 17.cronjob that initiates collection of data and send it to the web server.

```
crontab -l -u pi

# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

*/2   * * * *  /trap/read_sensors.py
*/30  * * * *  /trap/upload_sensor_data.py
```

The two python scripts a) collect sensor values and store them in a file in /trap/buffer folder and b) reads contents from the folder /trap/buffer and uploads them to the webserver using POST with JSON data.

Listing 18. Python script to read sensor values and store them in files "**read_sensors.py**"

```
#!/usr/bin/env python3
#This python script reads sensor values and stores them in a text file with
extension json.dumps, in folder /trap/buffer
import errno
import os
import subprocess
from datetime import datetime
import urllib.request
import json
import RPi.GPIO as GPIO
import time
import base64

GPIO.setmode(GPIO.BCM)
GPIO.setup(20,GPIO.OUT)
```

```
GPIO.setup(21,GPIO.OUT)

# GSM does not work so no need to turn on and off
#print ("LED 20 ON (GSM/GPRS Modem)")
#GPIO.output(20,GPIO.HIGH)
#time.sleep(1)
#print("LED 20 off")
#GPIO.output(20,GPIO.LOW)

# Turn on power to anemometer to read wind speed
print ("21 ON (LED Strip - Anemometer)")
GPIO.output(21,GPIO.HIGH)

debug=0
my_timestamp       = datetime.now().strftime('%Y-%m-%d_%H-%M-%S')
my_year_month_path = datetime.now().strftime('%Y/%m')
my_filename = '/trap/buffer/'+my_timestamp+'_SensorDat.json.dumps'
print(my_timestamp)
print(my_filename)

#############################################################################
# FUNCTIONS
#############################################################################
def create_path(path):
    try:
       os.makedirs(path)
    except OSError as exception:
       if exception.errno != errno.EEXIST:
           raise

def filecreation(list, filename):
    mydir = os.path.join(
        #os.getcwd(),                # get current directory
        "/trap/buffer")
    try:
        os.makedirs(mydir, exist_ok=True)
    except OSError as e:
        if e.errno != errno.EEXIST:
            raise  # This was not a "directory exist" error..

    try:
        with open(os.path.join(mydir, filename), 'w') as d:
            d.writelines(list)
    except OSError as e:
            print(e.errno)
            raise

#############################################################################
# MAIN PROGRAM
#############################################################################
# Variable Default Values
trapID='Dacus-CRCH0001-0001'
webserverHASH='ADC235A456111123'
Humidity=0
Temperature=0
BaromerticPressure=0
BatteryVoltage=12.1
WindSpeed=3
broadband=0
ir=0
```

```
lux=0
insectsFound=0

# Get Humidity and Temperature from AM2315
tmpAM2315=subprocess.getoutput('/trap/AM2315/read')
print(tmpAM2315)
Humidity=tmpAM2315.split(',',1)[0].split(':',1)[1].strip().split('
',1)[1].split('%',1)[0]
Temperature=tmpAM2315.split(',',1)[1].split(':',1)[1].strip()

# Get Temperature and Barometric Pressure from BMP280
tmpBMP280=subprocess.getoutput('/trap/bmp280/bmp280')
print(tmpBMP280)
BaromerticPressure=tmpBMP280.split(',',1)[0].split(':',1)[1].strip().split('
',1)[0]
Temperature1=tmpBMP280.split(',',1)[1].split(':',1)[1].strip().split(' ',1)[0]

# Get lighting broadband, ir, lux  from Light Sensor
tmpTSL2561=subprocess.getoutput('/trap/lux/TSL2561_test')
print(tmpTSL2561)
broadband=tmpTSL2561.split(',')[1].split(':',1)[1].strip()
ir=tmpTSL2561.split(',')[2].split(':',1)[1].strip()
lux=tmpTSL2561.split(',')[3].split(':',1)[1].strip()

# Get windspeed and battery voltage
tmpADC=subprocess.getoutput('/trap/ads1015/Singleended')
print(tmpADC)
BatteryVoltage=round(int(tmpADC.split(',')[1].split(':',1)[1].strip())/115,2)
WindSpeed=round((((int(tmpADC.split(',')[2].split(':',1)[1].strip())/1000)-
0.4)/1.6)*32.4,2)


# Read image output from Machine Vision processing.
# /trap/cpp/grab_removeHole_histEq_OtsuThresh --help
# Found:3, image:/trap/grabbed/ID001__blur_11xOtsu_elipses_2017_06_10__12-43-
36.jpg
#
# --timestamp <fname>: File to write to
# --path <fullpath>:   Path to write files [default=/trap/grabbed]
path2="grabbed/"+my_year_month_path
create_path("/trap/"+path2)
tmpOpenCV=subprocess.getoutput('/trap/cpp/grab_removeHole_histEq_OtsuThresh --
path='+path2+' --id='+trapID+' --timestamp='+my_timestamp)
print(tmpOpenCV)
insectsFound=int(tmpOpenCV.split(',')[0].split(':',1)[1].strip())
jpegImage=tmpOpenCV.split(',')[1].split(':',1)[1].strip()

encodedImage = base64.b64encode(open(jpegImage, "rb").read())
str_encodedImage=encodedImage.decode('utf-8')

print("21 off")
GPIO.output(21,GPIO.LOW)


body = {'TrapID':str(trapID), 'webserverHASH':str(webserverHASH),
'ts':str(my_timestamp),'v':str(BatteryVoltage),'t':str(Temperature),'h':str(Humid
ity),'p':str(BaromerticPressure),'ws':str(WindSpeed),'lbb':str(broadband),'lux':s
tr(lux),'lir':str(ir),'insectCountOCV':str(insectsFound),'img':str_encodedImage}

#save json data to buffer_file
```

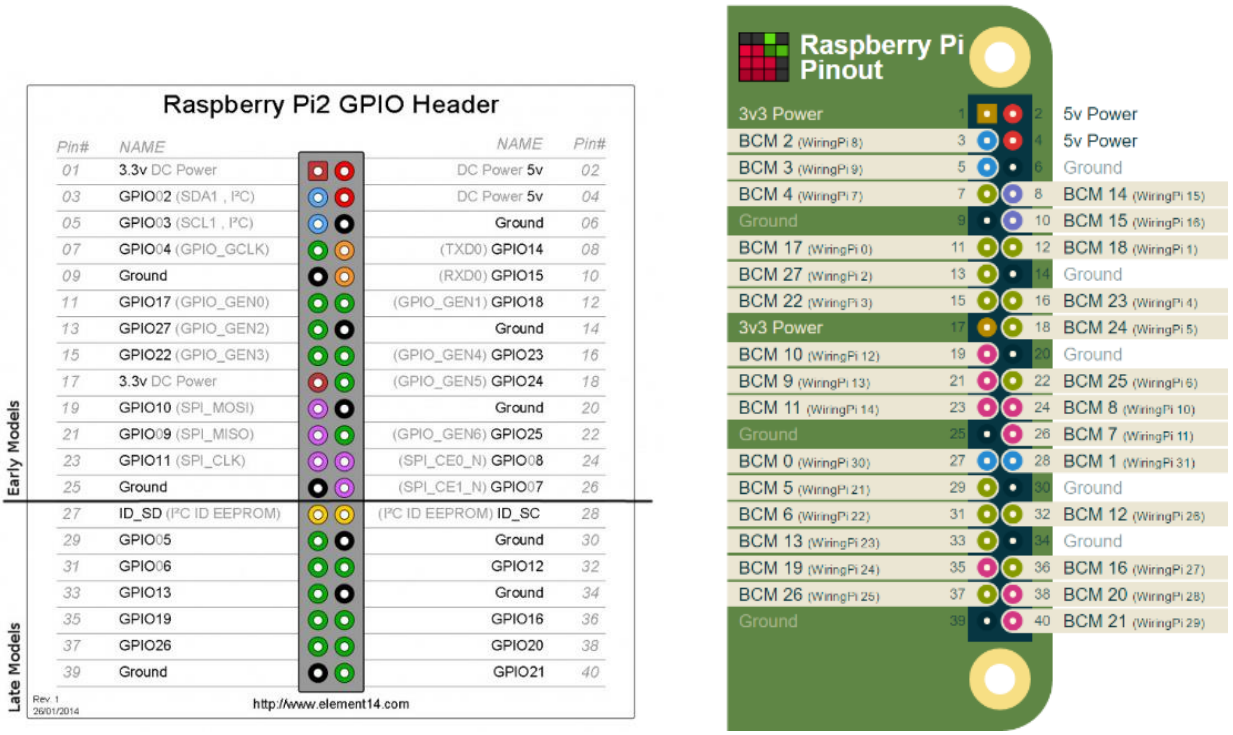
```
with open(my_filename, mode='w', encoding='utf-8') as a_file:
    a_file.write(json.dumps(body))

if (debug):
       myurl = "https://thesis.karapatakis.gr/api_data"
       req = urllib.request.Request(myurl)
       req.add_header('Content-Type', 'application/json; charset=utf-8')
       jsondata = json.dumps(body)
       jsondataasbytes = jsondata.encode('utf-8')   # needs to be bytes
       req.add_header('Content-Length', len(jsondataasbytes))
       print (jsondataasbytes)
       response = urllib.request.urlopen(req, jsondataasbytes)
       content =  response.read().decode(response.headers.get_content_charset())
       print("Response from POST API: "+content)
```

Listing 19. Python script to upload data to the webserver "upload_sensor_data.py"

```
#!/usr/bin/env python3

#This python script finds files with filename ending in json.dumps, reads them
one by one into a variable, sends the data to the webserver and then moves the
file to a different folder named “processed”

import errno
import os
import subprocess
from datetime import datetime
import urllib.request
import json
import RPi.GPIO as GPIO
import time
import base64
import glob
import shutil

myurl = "https://thesis.karapatakis.gr/api_data"
req = urllib.request.Request(myurl)
req.add_header('Content-Type', 'application/json; charset=utf-8')
#read json data from file to a variable.
my_files=glob.glob('/trap/buffer/*.json.dumps')
for jsonFile in my_files:
    with open(jsonFile, mode='r', encoding='utf-8') as a_file:
        #body=json.reads(a_file.read())
        jsondata=a_file.read()
        print(jsondata)
        #jsondata = json.dumps(body)
        jsondataasbytes = jsondata.encode('utf-8')   # needs to be bytes
        req.add_header('Content-Length', len(jsondataasbytes))
#print (jsondataasbytes)
        #response = urllib.request.urlopen(req, jsondataasbytes)
        #content =
response.read().decode(response.headers.get_content_charset())
        #print("Response from POST API: "+content)
    #if all is good _ move file to processed_directory
    shutil.move(jsonFile,'/trap/buffer/processed/')
fi
```

APPENDIX B: GPIO Pins, usage, naming conventions

The raspberry pi3 has 40 GPIO pins, some pins are General Purpose Input / Output, and some cannot be used, because they are used for example for the I2C, or SPI bus, TR, RX or they just provide 3.3V, 5V and GND.



a) GPIO Header Pinout b) Pinouts when using the WiringPi library

Figure 54. Raspberry Pi GPIO Header Pinout

APPENDIX C: Troubleshooting, errors

Timing Errors when reading sensors with data pins

In the early stages of implementation, we used the DHT11 Temperature and Humidity sensor to test the environmental monitoring aspects of the device. The sensor was connected to a GPIO data pin, and we allowed the system to run, collecting data for well over 6 days, making well over 260.000 attempts to read data from the sensor.

The realization we arrived at was, that in about 20.12% of the time data that was read was invalid. Since the parameters we monitor are not subject to rapid changes, and are not used to control critical processes e.g. power plants, the decisions made were a. to poll sensors for data less frequently b. in order to accurately and read sensor data the first time we poll them we will use I2C bus connected sensors.

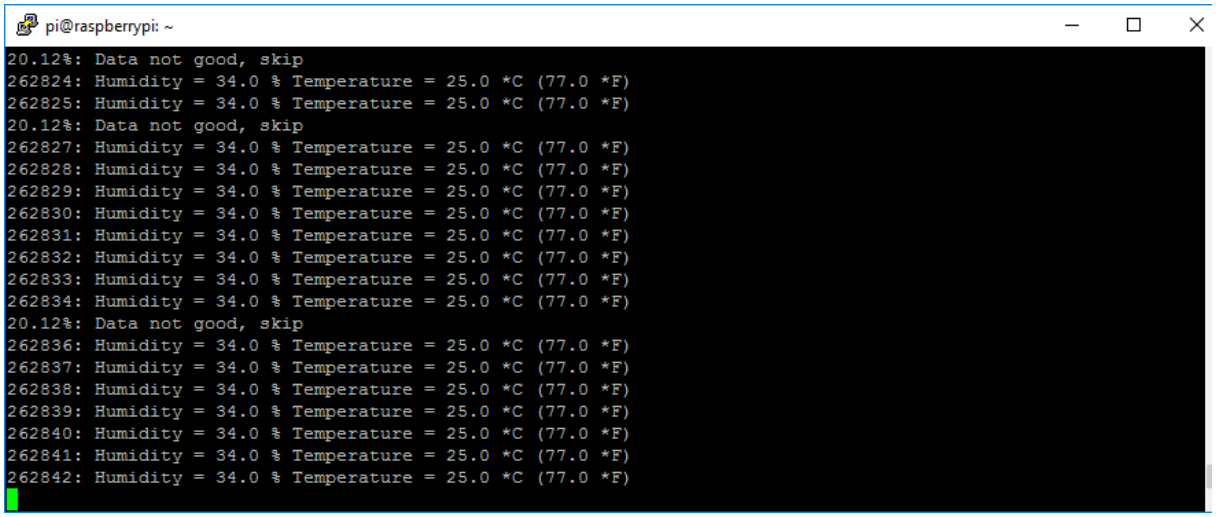


Figure 55. Raspberry Pi, reading the temperature and humidity from a DHT11 sensor

Error reading sensors connected to the I2C bus

Although the AOSONG encased I2C Temperature/Humidity Sensor (AM2315) was connected to the I2C bus with the recommended from the company 10K pullup resistors, it does not seem to function properly. It is detected only half of the time, some other users have reported the same behavior (Sopwith, 2014) and determined that the sensor is asleep (low power mode) most of the time as a measure to reduce the likelihood of heat affecting the Humidity sensor reading.

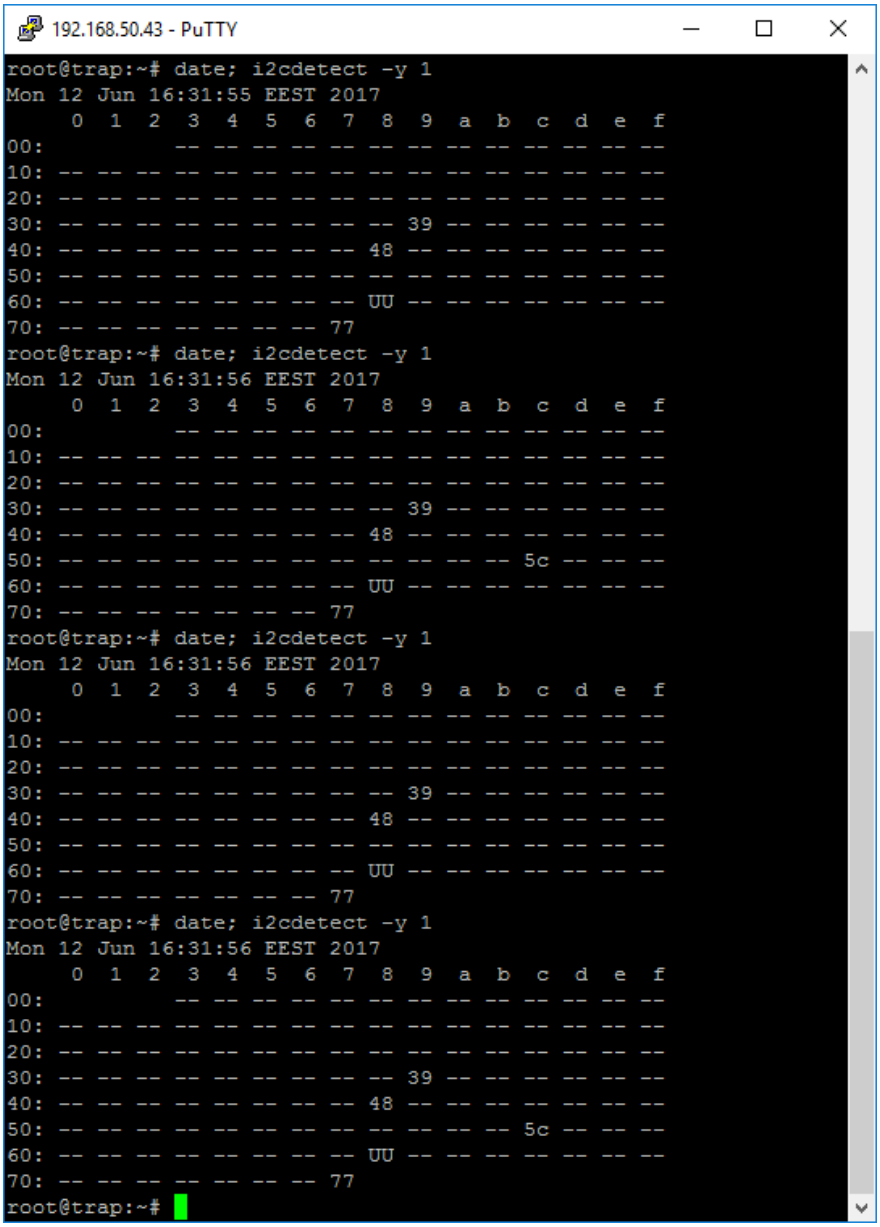


Figure 56. i2c bus seems not to detect the sensor with address 5c (AM2315)

Error connecting to the Vodafone GSM/GPRS Network

Sending AT commands to the GSM modem worked without a problem, but when we want to create a pppd profile and set it up so the Linux Operating System can initiate and make an internet connection when needed.

After talking to a Vodafone (Greece) Technician, I was informed that they could not do any debugging from their side, and pointed out that the only configuration needed was:

APN	internet
Dial String	ATD*99#

and if authentication if required the default PAP username and password are:

username	user
password	pass

The configuration files used are in the following listings.

Listing 20. ppp profile /etc/ppp/peer/fona

```
# Example PPPD configuration for FONA GPRS connection on Debian/Ubuntu.

# MUST CHANGE: Change the -T parameter value to your network's APN value.
# For example if your APN is 'internet', the line looks like:
# connect "/usr/sbin/chat -v -f /etc/chatscripts/gprs -T internet"

# for vodafone Greece, with a internet only SIM, APN = internet
connect "/usr/sbin/chat -v -f /etc/chatscripts/gprs -T internet"

# MUST CHANGE: Uncomment the appropriate serial device for your platform.
# For Raspberry Pi use /dev/ttyAMA0 by uncommenting the line below:
#/dev/ttyAMA0
# For BeagleBone Black use /dev/ttyO4 by uncommenting the line below:
#/dev/ttyO4
/dev/serial0

# Speed of the serial line.
115200

# Assumes that your IP address is allocated dynamically by the ISP.
noipdefault

# Try to get the name server addresses from the ISP.
usepeerdns

# Use this connection as the default route to the internet.
defaultroute

# Makes PPPD "dial again" when the connection is lost.
persist

# Do not ask the remote to authenticate.
noauth

# No hardware flow control on the serial link with FONA
```

```
nocrtscts
# No modem control lines with FONA.
local
#nodeflate
debug
```

Listing 21. chatscript to initiate gprs: /etc/chatscripts/gprs

```
# You can use this script unmodified to connect to cellular networks.
# The APN is specified in the peers file as the argument of the -T command
# line option of chat(8).

# For details about the AT commands involved please consult the relevant
# standard: 3GPP TS 27.007 - AT command set for User Equipment (UE).
# (http://www.3gpp.org/ftp/Specs/html-info/27007.htm)
ABORT       BUSY
ABORT       VOICE
ABORT       "NO CARRIER"
ABORT       "NO DIALTONE"
ABORT       "NO DIAL TONE"
ABORT       "NO ANSWER"
ABORT       "DELAYED"
ABORT       "ERROR"

# cease if the modem is not attached to the network yet
ABORT       "+CGATT: 0"

""      AT
TIMEOUT     12
OK      ATH
OK      ATE1
OK      AT+CPIN?
OK      AT+COPS=?
OK      at+cops?
OK      at+cgreg?

# +CPIN provides the SIM card PIN
#OK     "AT+CPIN=1234"

# +CFUN may allow to configure the handset to limit operations to
# GPRS/EDGE/UMTS/etc to save power, but the arguments are not standard
# except for 1 which means "full functionality".
#OK     AT+CFUN=1

#OK     AT+CGDCONT=1,"IP","\T","",0,0
OK      ATD*99#
TIMEOUT     22
CONNECT     "" #nodeflate
```

Despite our efforts the modem could not connect to the network, and the log files after enabling debug showed that the error reported was: Protocol-Reject for 'Compression Control Protocol' (0x80fd) received.

Detailed logs are in the following listing:

Listing 22. Errors found in syslog: /var/log/syslog

```
root@trap:/etc# Jun 20 07:13:01 trap chat[18489]: abort on (BUSY)
Jun 20 07:13:01 trap chat[18489]: abort on (VOICE)
Jun 20 07:13:01 trap chat[18489]: abort on (NO CARRIER)
Jun 20 07:13:01 trap chat[18489]: abort on (NO DIALTONE)
```

```
Jun 20 07:13:01 trap chat[18489]: abort on (NO DIAL TONE)
Jun 20 07:13:01 trap chat[18489]: abort on (NO ANSWER)
Jun 20 07:13:01 trap chat[18489]: abort on (DELAYED)
Jun 20 07:13:01 trap chat[18489]: abort on (ERROR)
Jun 20 07:13:01 trap chat[18489]: abort on (+CGATT: 0)
Jun 20 07:13:01 trap chat[18489]: send (AT^M)
Jun 20 07:13:01 trap chat[18489]: timeout set to 12 seconds
Jun 20 07:13:01 trap chat[18489]: expect (OK)
Jun 20 07:13:01 trap chat[18489]: AT^M^M
Jun 20 07:13:01 trap chat[18489]: OK
Jun 20 07:13:01 trap chat[18489]:  -- got it
Jun 20 07:13:01 trap chat[18489]: send (ATH^M)
Jun 20 07:13:01 trap chat[18489]: expect (OK)
Jun 20 07:13:01 trap chat[18489]: ^M
Jun 20 07:13:01 trap chat[18489]: ATH^M^M
Jun 20 07:13:01 trap chat[18489]: OK
Jun 20 07:13:01 trap chat[18489]:  -- got it
Jun 20 07:13:01 trap chat[18489]: send (ATE1^M)
Jun 20 07:13:01 trap chat[18489]: expect (OK)
Jun 20 07:13:01 trap chat[18489]: ^M
Jun 20 07:13:01 trap chat[18489]: ATE1^M^M
Jun 20 07:13:01 trap chat[18489]: OK
Jun 20 07:13:01 trap chat[18489]:  -- got it
Jun 20 07:13:01 trap chat[18489]: send (AT+CPIN?^M)
Jun 20 07:13:01 trap chat[18489]: expect (OK)
Jun 20 07:13:01 trap chat[18489]: ^M
Jun 20 07:13:01 trap chat[18489]: AT+CPIN?^M^M
Jun 20 07:13:01 trap chat[18489]: +CPIN: READY^M
Jun 20 07:13:01 trap chat[18489]: ^M
Jun 20 07:13:01 trap chat[18489]: OK
Jun 20 07:13:01 trap chat[18489]:  -- got it
Jun 20 07:13:01 trap chat[18489]: send (AT+COPS=?^M)
Jun 20 07:13:01 trap chat[18489]: expect (OK)
Jun 20 07:13:01 trap chat[18489]: ^M
Jun 20 07:13:11 trap chat[18489]: AT+COPS=?^M^M
Jun 20 07:13:11 trap chat[18489]: +COPS: (2,"vodafone","voda
GR","20205"),(3,"TIM","TIM","20210"),(3,"COSMOTE","C-
Jun 20 07:13:11 trap chat[18489]: OTE","20201"),,(0-4),(0-2)^M
Jun 20 07:13:11 trap chat[18489]: ^M
Jun 20 07:13:11 trap chat[18489]: OK
Jun 20 07:13:11 trap chat[18489]:  -- got it
Jun 20 07:13:11 trap chat[18489]: send (at+cops?^M)
Jun 20 07:13:11 trap chat[18489]: expect (OK)
Jun 20 07:13:11 trap chat[18489]: ^M
Jun 20 07:13:11 trap chat[18489]: at+cops?^M^M
Jun 20 07:13:11 trap chat[18489]: +COPS: 0,0,"vodafone"^M
Jun 20 07:13:11 trap chat[18489]: ^M
Jun 20 07:13:11 trap chat[18489]: OK
Jun 20 07:13:11 trap chat[18489]:  -- got it
Jun 20 07:13:11 trap chat[18489]: send (at+cgreg?^M)
Jun 20 07:13:12 trap chat[18489]: expect (OK)
Jun 20 07:13:12 trap chat[18489]: ^M
Jun 20 07:13:12 trap chat[18489]: at+cgreg?^M^M
Jun 20 07:13:12 trap chat[18489]: +CGREG: 0,1^M
Jun 20 07:13:12 trap chat[18489]: ^M
Jun 20 07:13:12 trap chat[18489]: OK
Jun 20 07:13:12 trap chat[18489]:  -- got it
Jun 20 07:13:12 trap chat[18489]: send (ATD*99#^M)
Jun 20 07:13:12 trap chat[18489]: timeout set to 22 seconds
Jun 20 07:13:12 trap chat[18489]: expect (CONNECT)
```

```
64  Jun 20 07:13:12 trap chat[18489]: ^M
65  Jun 20 07:13:12 trap chat[18489]: ATD*99#^M^M
66  Jun 20 07:13:12 trap chat[18489]: CONNECT
67  Jun 20 07:13:12 trap chat[18489]:  -- got it
68  Jun 20 07:13:12 trap chat[18489]: send (^M)
69  Jun 20 07:13:12 trap pppd[18481]: Script /usr/sbin/chat -v -f
70  /etc/chatscripts/gprs -T internet finished (pid 18488), status = 0x0
    Jun 20 07:13:12 trap pppd[18481]: Script /usr/sbin/chat -v -f
71  /etc/chatscripts/gprs -T internet finished (pid 18488), status = 0x0
72  Jun 20 07:13:12 trap pppd[18481]: Serial connection established.
73  Jun 20 07:13:12 trap pppd[18481]: using channel 138
74  Jun 20 07:13:12 trap pppd[18481]: Using interface ppp0
75  Jun 20 07:13:12 trap pppd[18481]: Connect: ppp0 <--> /dev/serial0
76  Jun 20 07:13:13 trap pppd[18481]: rcvd [LCP ConfReq id=0x1 <asyncmap 0xa0000>
    <auth pap> <pcomp> <accomp>]
77  Jun 20 07:13:13 trap pppd[18481]: sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic
    0xaac9c774> <pcomp> <accomp>]
78  Jun 20 07:13:13 trap pppd[18481]: sent [LCP ConfAck id=0x1 <asyncmap 0xa0000>
    <auth pap> <pcomp> <accomp>]
79  Jun 20 07:13:13 trap pppd[18481]: rcvd [LCP ConfNak id=0x1 <asyncmap 0xa0000>]
80  Jun 20 07:13:13 trap pppd[18481]: sent [LCP ConfReq id=0x2 <asyncmap 0xa0000>
    <magic 0xaac9c774> <pcomp> <accomp>]
81  Jun 20 07:13:13 trap pppd[18481]: rcvd [LCP ConfAck id=0x2 <asyncmap 0xa0000>
    <magic 0xaac9c774> <pcomp> <accomp>]
82  Jun 20 07:13:13 trap pppd[18481]: sent [PAP AuthReq id=0x1 user="trap"
    password=<hidden>]
83  Jun 20 07:13:13 trap pppd[18481]: rcvd [PAP AuthAck id=0x1 ""]
84  Jun 20 07:13:13 trap pppd[18481]: PAP authentication succeeded
85  Jun 20 07:13:13 trap pppd[18481]: sent [CCP ConfReq id=0x1 <bsd v1 15>]
86  Jun 20 07:13:13 trap pppd[18481]: sent [IPCP ConfReq id=0x1 <compress VJ 0f 01>
    <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
87  Jun 20 07:13:13 trap pppd[18481]: rcvd [IPCP ConfReq id=0x1]
88  Jun 20 07:13:13 trap pppd[18481]: sent [IPCP ConfNak id=0x1 <addr 0.0.0.0>]
89  Jun 20 07:13:13 trap pppd[18481]: rcvd [IPCP ConfReq id=0x2 <addr 0.0.0.0>]
90  Jun 20 07:13:13 trap pppd[18481]: sent [IPCP ConfRej id=0x2 <addr 0.0.0.0>]
91  Jun 20 07:13:13 trap pppd[18481]: rcvd [IPCP ConfReq id=0x3]
92  Jun 20 07:13:13 trap pppd[18481]: sent [IPCP ConfAck id=0x3]
93  Jun 20 07:13:16 trap pppd[18481]: sent [CCP ConfReq id=0x1 <bsd v1 15>]
94  Jun 20 07:13:16 trap pppd[18481]: sent [IPCP ConfReq id=0x1 <compress VJ 0f 01>
    <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
95  Jun 20 07:13:16 trap pppd[18481]: rcvd [LCP ProtRej id=0x0 80 fd 01 01 00 07 15
    03 2f]
96  Jun 20 07:13:16 trap pppd[18481]: Protocol-Reject for 'Compression Control
    Protocol' (0x80fd) received
97  Jun 20 07:13:16 trap pppd[18481]: rcvd [IPCP ConfRej id=0x1 <compress VJ 0f 01>]
98  Jun 20 07:13:16 trap pppd[18481]: sent [IPCP ConfReq id=0x2 <addr 0.0.0.0> <ms-
    dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
99  Jun 20 07:13:16 trap pppd[18481]: rcvd [LCP TermReq id=0x2]
100 Jun 20 07:13:16 trap pppd[18481]: LCP terminated by peer
101 Jun 20 07:13:16 trap pppd[18481]: sent [LCP TermAck id=0x2]
102 Jun 20 07:13:19 trap pppd[18481]: Connection terminated.
103 Jun 20 07:13:19 trap pppd[18481]: Modem hangup
```

We suspect either some hardware malfunction, incompatibility or other configuration error that does not allow us to connect to the 3G Network.

Computer Vision, bad design ideas

During the development of the software for the machine vision algorithm, not everything was easy even when everything seemed to be working perfectly, unexpected errors hindered our progress. Even when the algorithm seemed to work as expected, Figure 57. Machine vision algorithm to remove artifacts from the center hole of the trap, when we tries to automatically find the region of interest, relying on the color, it could not be done easily.

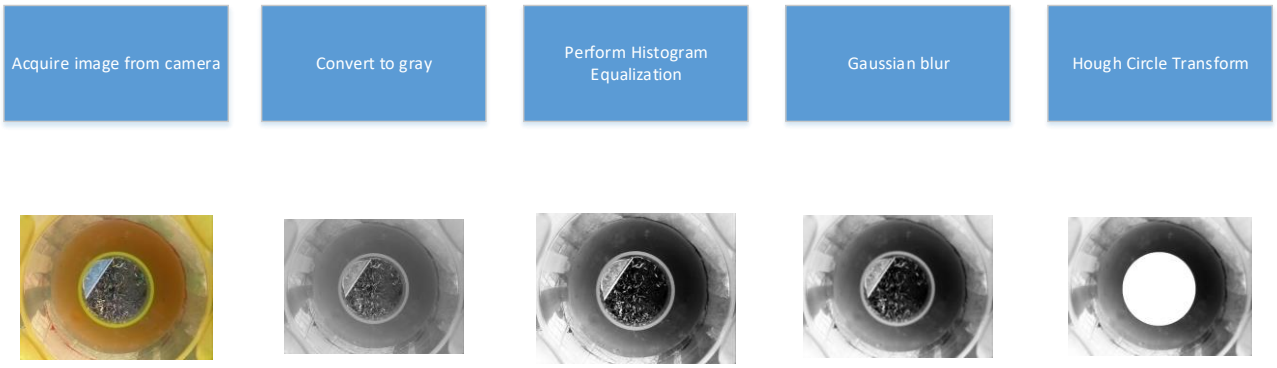


Figure 57. Machine vision algorithm to remove artifacts from the center hole of the trap

To overcome the problem an approach to convert the acquired images to the HSV color space was tested to try to isolate Regions of Interest, with varying amount of success where lighting conditions, liquid movement and reflections from light sources created problems.

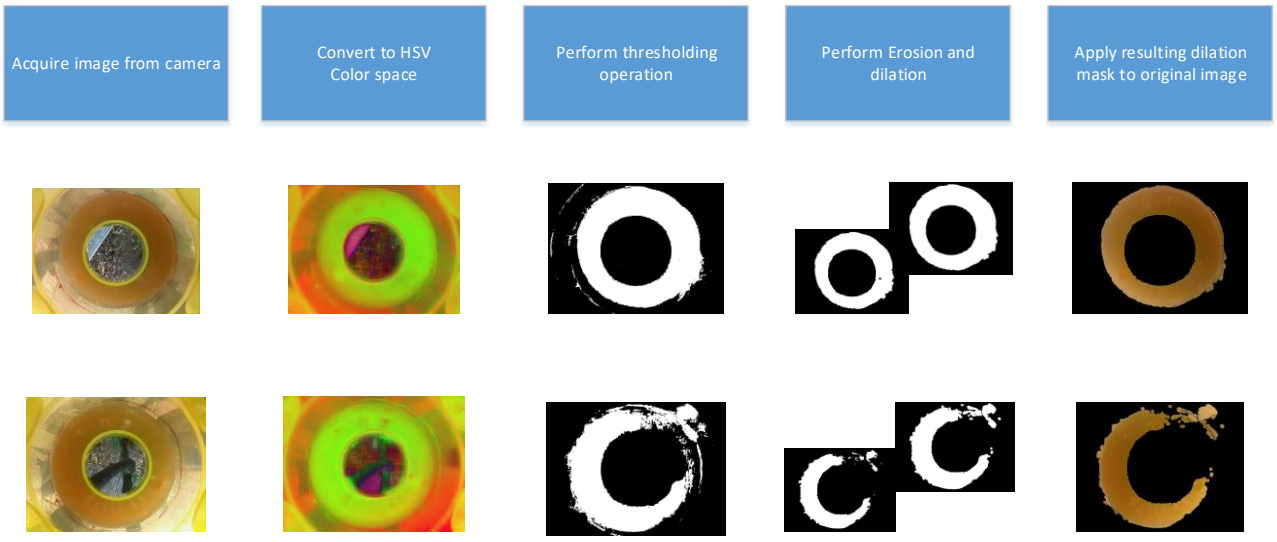


Figure 58. machine vision algorithm to detect liquid surface using HSV color space yields inconsistent results

In the figure above two experiments show that segmenting the liquid from the trap and background is not that easy, and cannot properly segmented every time in a robust way.

C++ runtime errors (Segmentation Faults)

During the development of the software for the machine vision algorithm, not everything was easy even when everything seemed to be working perfectly, unexpected errors hindered our progress. After most of the development was finished, some random execution errors emerged. Programs that previously run without a problem, crashed reporting segmentation fault.

In the process to figure out what caused the problem in the source code, the program was recompiled with the debug flag **`-g`** and using the instructions about **`gdb`** debugging (Stack Overflow Community, 2013) the problem was isolated.

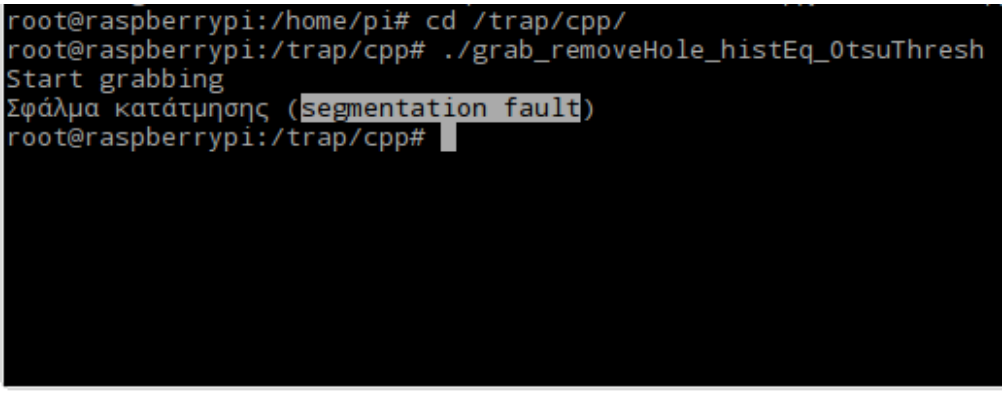


Figure 59. segmentation fault when executing main machine vision program

APPENDIX D: Materials used

Hardware cost breakdown for materials used

The total cost of the system is around €230 using off the shelf electronics. The breakdown of the individual hardware components cost for the whole project is given in detail in the following table.

The hardware costs are reasonable depending on the number of sensors we install on each trap. For a given geographical region only one trap need have environmental sensors, thus reducing the overall cost since all other McPhail traps would only have the vision sensors.

Item	Model	Current price €
Raspberry Mini Computer	Raspberry Pi3	35
Pi CSI Camera Cable 2m	Flex cable for Raspberry Pi Camera 2m	4
GPIO Breakout for Raspberry Pi	Adafruit Accessories Assembled Pi T-Cobbler Plus - GPIO Breakout for RasPi A+/B+/Pi 2 (1 piece)	10
Raspberry Pi Case	Raspberry Pi 3 Official case (plastic)	6
Raspberry Pi Camera	Camera Module Board 5MP 160° Wide Angle Fish Eye Lenses for Raspberry Pi	23
Temperature and Barometric Pressure sensor	Adafruit BMP280 I2C/SPI Barometric pressure	10
Temperature and Humidity sensor	AM2315-Encased I2C Temperature / Humidity	24
Anemometer/Wind sped sensor	Adafruit Anemometer Wind Sped sensor with analog voltage output	45
Digital Luminosity, Lux, Light sensor	TSL2561 Digital Luminosity/LUX/LIGHT Sensor	6
Analog to Digital Converter	ADS1015 12-Bit ADC, 4 CHANNEL WITH PRO	7
Real Time Clock	Adafruit PCF8523 RTC	5
GSM Modem SIM 800	SainSmart SIM800 850/900/1800/1900 MHz GPRS/GSM Board Quad-Band Module Kit	20
Voltage Regulator for GSM Modem	12-30V input to variable output voltage regulator for	10
12V Car USB Charger	12 Volt cigarette lighter USB charger	5
12V Battery	12V Car Battery (used)	20