DISSERTATION

University of Southern Queensland

Faculty of Health, Engineering and Sciences

AUTOMATED IDENTIFICATION OF INSECT PESTS IN TRAPS

A dissertation submitted by

Pierre Rousseau

in fulfilment of the requirements of

ENG4111 and 4112 Research Project

towards the degree of

Bachelor of Engineering (Honours) (Mechatronic)*

Submitted October, 2020

Abstract

Insect pests are a significant risk to the agricultural and horticultural sectors and present a major risk to many billions of dollars' worth of food production. Significant infestations can threaten food security, as well as export markets due to importing countries placing restrictions on a region of origin.

This study investigated the Mediterranean fruit fly (Medfly), species name Ceratitis capitata (Wiedemann). Fruit flies threaten a $12 billion industry and are widespread. Medfly is the most ubiquitous of the fruit fly pest species and therefore poses a significant threat to agriculture worldwide.

The objectives of the project were to develop a low-cost and open source automated fruit fly trap, which could be used for remote monitoring of fruit fly traps. Due to various limitations the system could not be implemented as initially envisaged but a partially functional system was achieved.

Some system components were successfully or partially implemented. Construction and remote operation of a Bosch BME280 temperature, pressure and relative humidity sensor, along with provision for data logging for this sensor was implemented. This system was successfully operated remotely via a WiFi connection. A 16MP camera was partially implemented but failed to perform as expected and further work on this hardware was abandoned. An Adafruit neopixel 12 LED ring was successfully implemented, to be used as a light source for image collection in the dark. Although the camera did not perform as expected, partial success was achieved in processing images by discriminating Medflies within a trap using the OpenCV Simple Blob Detector and saving the fly count to a csv file. No other easily implementable image processing options were found in OpenCV, which could discriminate fruit flies from background. A control algorithm was also developed to drive all the system components and to email a csv file with collated weather data and fly counts, with time stamps, to a designated email address. A housing for the entire system was designed and 3D printed at USQ and a partially operational prototype was constructed.

The viability of the project is still considered to be feasible, although the work required and the challenges involved to achieve completion necessitate that much more time will need to be spent on the project to achieve a fully functional model, with potential for many years of further development.

Obtaining a better camera and further development of the fruit fly identification software, power management, communication protocols and lighting system are all considered to be important.

University of Southern Queensland

Faculty of Health, Engineering and Sciences

ENG4111 & ENG4112 Research Project

Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Health, Engineering and Sciences, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Health, Engineering and Sciences or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitles “Research Project” is to contribute to the overall education within the student’s chosen degree program. This document, the associated hardware, software, drawings, and any other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Certification

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

[redacted]

Pierre Rousseau

Student Number: [redacted]

Acknowledgements

Thank you to Dr. Craig Lobsey for suggesting this project and acting as my project supervisor and for his valuable advice and support.

Thank you to Mr. Terry Byrne and Mr. Graham Holmes for advice on improving the equipment housing design and their assistance with 3D printing it.

Thank you to Dr. Sandra Broughton, Dr. Vineeta Bilgi and their colleagues at the Department of Primary Industries and Regional Development - Western Australia (Agriculture and Food – Biosecurity) for supplying fruit fly samples used for testing. These samples helped tremendously in progressing the project.

I would also like to thank my wife, Dr. Lauren Pham, for her ongoing patience and support throughout the time this course has taken to complete and for proofreading this dissertation. It is greatly appreciated.

Also thanks to my family who supported me throughout this time.

Contents

Figures

Tables

Appendices

1 Background

Insect pests are a major cause of agricultural losses, with between 10% and 16% of produce lost during production and a similar amount after harvest. The spread of pests is primarily facilitated by human transportation. Major famines have been caused through the destruction of staple crops by pests (Bebber, Ramotowski & Gurr 2013).

There are many locally significant and potentially threatening exotic insect pests in Australia, as observed by various agencies (Department of Agriculture and Fisheries - Queensland Government 2017, Department of Primary Industries and Regional Development - Western Australia n.d., Department of Agriculture, Water and the Environment - Australian Government 2019 and Australian Interstate Quarantine 2019).

Many insect pests are identified as posing a risk to agriculture and horticulture. Some of the potentially high-risk insects are limited to specific regions and / or do not have a long history of monitoring in Australia. The fall army worm is such an example. It has only been detected in the York Peninsula since January 2020 and in Kununarra (Western Australia) in March 2020, yet is considered to be high risk and has been designated a declared pest (Department of Primary Industries and Regional Development - Western Australia 2020). It was not considered to be a viable research subject due to its inaccessibility from Perth, especially considering likely ongoing travel restrictions due to COVID-19 during the project period, as well as budgetary constraints.

In the fruit-growing industry certain species of Diptera: Tephritidae (fruit flies) are a major pest. About 35% of fruit fly species attack soft fruits and some vegetables. Several species are critically important pests to fruit crops (De Meyer et al. 2008). All the cited information sources list fruit flies as a major threat to fruit and certain vegetable crops. Exotic fruit flies are listed as No. 3 in the Top 40 exotic and unwanted plant pests for Australia (Department of Agriculture, Water and the Environment - Australian Government 2019). Species that are specified include Bactrocera dorsalis (Oriental fruit fly), Zeugodacus curcubitae (Melon fly) and Anastrepha ludens (Mexican fruit fly). Only two fruit fly species are listed as being currently significant pests in Australia, namely Mediterranean Fruit Fly (Medfly) *Ceratitis capitata* (Wiedemann) and the Queensland Fruit Fly (Qfly) *Bactrocera tryoni*. Medfly is an exotic pest that originated in sub-Saharan Africa (Rahman & Broughton 2019) and Qfly is native to Queensland (Broughton 2020).

In Western Australia the Medfly (see Table 1) has become established. It can affect more than 200 types of fruit and vegetables. It does not occur on the East Coast of Australia and is

reportable anywhere outside Western Australia (Australian Interstate Quarantine 2019). It is listed as a priority pest in Queensland (Business Queensland 2019).

The Queensland Fruit Fly (Qfly) *Bactrocera tryoni* (see Table 1) attacks a larger range of crops than Medfly. It is absent from Western Australia, South Australia and Tasmania but present in the other states and territories (Broughton 2020). Cases of Qfly have been reported in the Perth region and eradicated periodically (Department of Primary Industries and Regional Development 2020).

The potential losses attributable to fruit flies are very important to the fruit industry. De Meyer et al. (2008) reported that losses were US$ 910 million to the Californian fruit industry in 1986, excluding an eradication program of US$ 290 million; losses in Israel, Palestinian Territories and Jordan were estimated at US$ 192 million in 1997 and losses in Egypt were estimated at €190 million in 1997. The same authors and Hafi et al. (2013) quoted the cost of an eradication program of Papaya fruit fly (subsequently found to be the same species as Oriental fruit fly (International Plant Protection Convention 2014)) in northern Queensland as AU$ 34 million. Costs of over AU$ 100 million to government, farmers and exporters, with projected eradication costs of AU$55-65 over the subsequent years were reported (Allwood & Leblanc 1997) and (Hafi et al. 2013). This outbreak also closed markets to Australian produce due to bans that were put in place by importing countries (Hafi et al. 2013). Hence, along with direct damages, loss of market access is an important consideration in controlling fruit flies.

Crop losses of >90% for some circumstances were reported by Allwood & Leblanc (1997). Researchers mention that accurate loss estimates are difficult to measure and obtain but that they potentially make up a significant proportion of the food supply (Allwood & Leblanc 1997 and De Meyer et al. 2008).

In the 2018 financial year the Tasmanian government budgeted AU$ 8 million for fruit fly eradication and the Federal Government was also reported to have provided AU$ 20 million to this cause (Beavis 2018).

Exotic fruit flies are potentially even more destructive than Qfly or Medfly, as some species can survive over a greater range of climatic conditions and can sting fruit at an earlier stage of development (Hafi et al. 2013). The estimated gross value of horticultural products that would be susceptible to exotic fruit fly species was AU$2.1 billion in 2011-2012. Calculations of the potential economic benefit of eradication of a local incursion of exotic fruit fly found that a strategy of 95% probability of success would have a net present value (NPV) of more than AU$ 2700 million (2013 value) at a cost of eradication of about AU$ 90 million (Hafi et al. 2013). The produce potentially affected by exotic fruit flies was estimated to be about half of

the total, in a horticultural sector worth about AU$ 12 billion per year in 2016-2017 (Plant Health Australia n.d.).

A National Fruit Fly Strategy (NFFS) was implemented in 2010 (Plant Health Australia n.d.b) and in 2015 the National Fruit fly Council was established to control fruit fly nationally, as a driver to foster coordination and collaboration between stakeholders (Plant Health Australia n.d.). Substantial investment in such collaborative fruit fly management and information shows how important this pest is to Australian horticulture. Examples include collaboration between various management teams (The Goulburn Murray Valley (GMV) Regional Fruit Fly Group 2019 and Plant Health Australia 2020) and dedicated websites, e.g. (Prevent Fruit Fly 2020, Fruit Fly Identification Australia 2020 and Plant Health Australia n.d.).

Table 1 Relevant fruit fly species identification

Common Name	Distinguishing features	Photo
Oriental Fruit Fly *Bactrocera dorsalis*	• Adults about 7mm long. • Can infest several kinds of fruit while they are still hard green or immature. Most other fruit fly species do not infest immature fruit. • Clear wings. • Generally a black back and paler abdomen with distinctive T-shaped marking on the back.	 (Business Queensland 2019)  (Fruit Fly Identification Australia 2020)

Common Name	Distinguishing features	Photo
Mediterranean Fruit Fly (Medfly) *Ceratitis capitate* (Wiedemann) **(Business Queensland 2019)**	• About 3-5 mm long. • Torax (back) mottled with shiny and dull black and yellowish-white areas. • Abdomen (rear body section) yellowish brown to brown with 2 pale cross bands. • Wings are patterned with yellow, brown and black spots and bands.	 (Broughton 2018)
Queensland Fruit Fly (Qfly) *Bactrocera tryoni* **(Broughton 2020)**	• Approximately 6-8 mm long. • Red eyes with very short antennae. • Thorax is reddish-brown with yellow patches on the sides and back. • Abdomen is solid dark brown, legs lighter brown and wings clear.	 (Broughton 2020)  (Fruit Fly Identification Australia 2020)

Currently implementation of an Integrated Pest Management (IPM) strategy is considered to be the optimal method to protect crops from pests. This approach follows a holistic path to pest control, as opposed to practices such as intensive spraying with insecticide. Issues such

as insecticide resistance, killing of non-target species and environmental toxicity are all factors favouring this approach, along with cost savings associated with lower pesticide consumption.

Effective and rapid monitoring is an integral part of IPM strategy (CSIRO 2020) and early detection of a pest invasion can make elimination possible, where a delay would permit it to spread (van Driesche, Hoddle & Center 2008). The ability to replace periodic manual monitoring (typically through weekly inspections) by automated systems, reduces a large amount of labour required to check and record results for potentially large numbers of traps, especially in large farming operations. Automation allows for rapid and targeted responses. Monitoring data can be processed to provide daily or even real time warnings, depending on the system, as discussed in Section 3.2.

Modelling by Cros et al. (2002) indicated that the impact of monitoring frequency on limiting the spread of an infestation of mobile insects was minimal, although it was significant for slow moving infestations such as pathogens and weeds. The dispersal rate of sterilised male Medflies, relevant to application of the Sterile Insect Technique (SIT), has been studied in several cases. It was found by Meats & Smallridge (2007) that dispersal of sterilised male fruit flies from a point of release followed a Cauchy distribution with respect to trap distance vs. proportion recaptured. The bulk of flies were recaptured close to the point of release (generally within 100 m) and a smaller number were caught up to 9.5 km away. A study by Paranhos et al. (2010) found that dispersion after release occurred relatively rapidly, with most dispersal happening within a day of release. No studies of natural infestation rates could be found, which would be more appropriate for predicting monitoring efficacy under field conditions. It is possible that natural infestation rates would differ substantially from the abovementioned SIT studies, as a wild population would be more diffuse to begin with and would likely either recolonise a former infestation site from pupae remaining at the site or invade the new territory opportunistically, initially through the pioneering individuals observed to travel longer distances.

Aside from any possible effect on the rate of advance of an infestation, earlier detection would be advantageous in limiting crop damage and also for reducing breeding success, especially at times when breeding cycles are shorter.

1.1 Target insect selection

Due to considerations of economic significance, research interest, location, cost and convenience, as well as travel restrictions associated with the COVID-19 pandemic, the Medfly was considered to be an ideal research subject for this project. Aside from being an established pest, which would be accessible without long distance travel, it would be

sufficiently technically challenging to identify through an automated monitoring system. This is because it is relatively small (3 to 5 mm long (Business Queensland 2019)), yet large enough to obtain reasonable photographs with a standard camera of high enough resolution. Any detection technology that is developed for it is likely to be applicable or easily adaptable to larger insect pests and potentially smaller easily recognisable species, as long as they can be lured to a suitable trap with selective (pheromone) bait. Trapping standards, equipment and methods are well developed for monitoring fruit fly, which supports performance assessment and comparison as well.

2 Ethical considerations

Ethical concerns for this project were minor. It did not propose to use technology that is different to current standards of best practice with respect to animal cruelty concerns. Automation and a low-cost open source platform may reduce employment and business opportunities for insect monitoring but is likely to have an overall beneficial impact in the horticulture industry, particularly for farmers with fewer resources than large farming enterprises. Some additional waste will be generated over time but with the benefit of much less travel for inspection. The potential for recycling could also limit this impact. Overall the project is therefore considered to be beneficial.

3 Literature Review

3.1 Ceratitis capitate (Medfly)

Medfly is considered to be one of the most invasive pests worldwide, due to its tolerance for a wide range of climatic conditions. Although it originated in sub-Saharan Africa (Sciarretta et al. 2018 and Rahman & Broughton 2019) the ecological niches and potential geographical distributions of Medfly are large and it is therefore considered to be the most serious fruit fly pest internationally (De Meyer et al. 2008). Potential distribution was modelled by De Meyer et al. (2008) using the genetic algorithm for rule-set prediction (GARP) and principal components analysis (PCA) methods. The GARP method was found to be a better predictor than the PCA method. The study found that Medfly thrives in both tropical and sub-tropical environments, with its predicted potential range, according to the GARP method, shown in Figure 1.

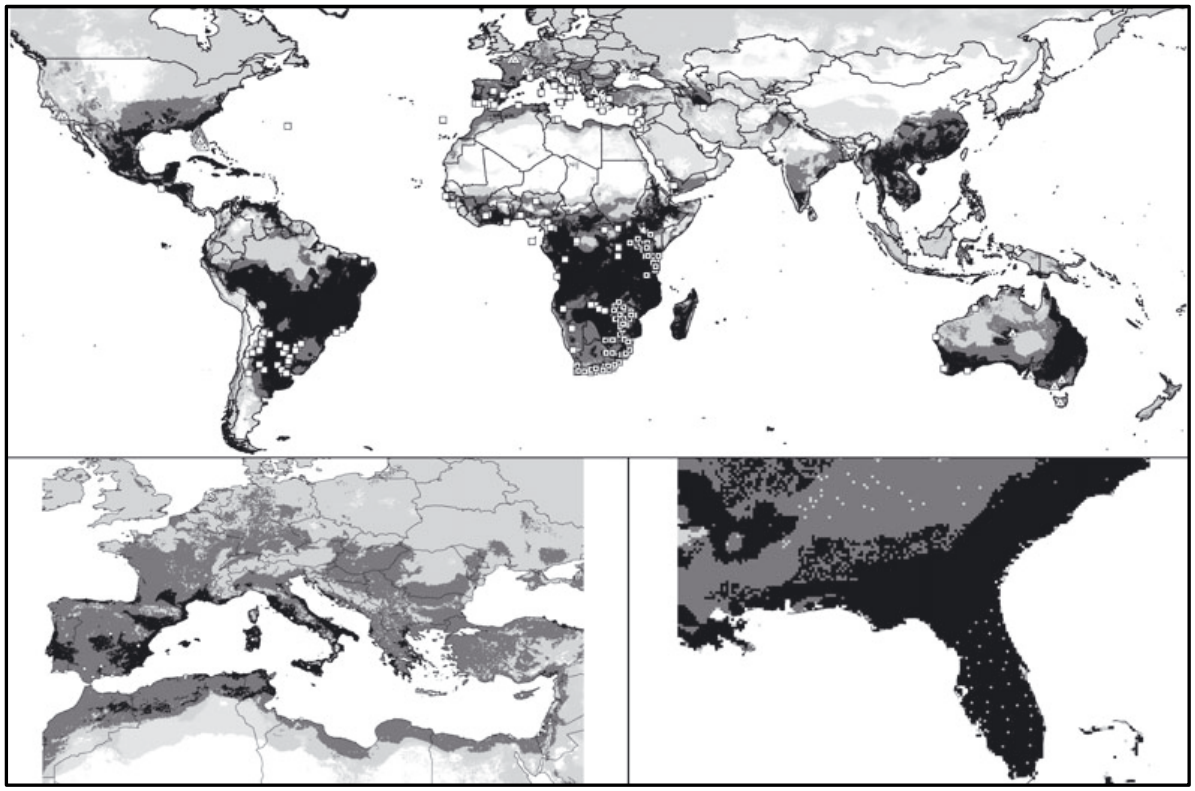


Figure 1 Potential Ceratitis capitata distribution according to GARP model (De Meyer et al. 2008).

The Medfly has been established in Western Australia for more than 100 years (first detected in Claremont in 1895 (Broughton 2018)) and affects most commercial fruit crops as well as 60 native fruit species. It was also reportedly present in New South Wales but was displaced by Qfly (Cook & Fraser 2014). Medfly are able to survive to near freezing temperatures for several weeks (De Lima et al. 2011 and Al-Behadili et al. 2019) and were found to survive in the Perth area of Western Australia over winter (Rahman & Broughton 2019).

The level of management of fruit flies depends on their breeding cycle, which is temperature dependent. Medflies become active when the temperature exceeds 12°C and complete a lifecycle in 28 - 34 days in summer and 60 – 115 days in winter (Broughton 2018). The rate of growth is linearly dependent on temperature in the range from 15°C to 30°C. Medfly is not tolerant of temperatures of 35°C or higher (Duyck & Quilici 2002). Formulas for growth rates for different stages of development at different constant temperatures are provided in Table 2, reformulated from linear regressions done by Duyck & Quilici (2002). Other factors such as Wolbachia infections and host fruit were also found to affect growth rates (Dionysopoulou et al. 2020).

Table 2 Formulas for calculating development times of Medfly lifestages, based on Duyck & Quilici (2002).

Life stage	Development time
Egg	$D = 100 / (3.5T - 41.2)$
Larva	$D = 100 / (1.1T - 11.5)$
Pupa	$D = 100 / (0.7T - 7.8)$
Ovarian maturation of adult females	$D = 100 / (1.1T - 9.9)$

D = Days for development, T = Temperature in °C

Cook & Fraser (2014) expected that increasing regulation of organophosphate pesticides would increase control costs in Western Australia and that eradication using the Sterile Insect Technique (SIT) would be more cost effective than ongoing control using bait sprays and intensive trapping. The SIT technique is currently favoured as an important fruit fly control strategy in Australia, as evidenced by its use in both Western Australia (Broughton 2019) and South Australia (PIRSA 2020).

The SIT technique relies on the use of gamma rays or X-rays to irradiate large numbers of males to sterilise them (Hendrichs & Robinson 2009), although genetically modified flies have also been trialled (Palmer 2017). Sterile males are released to mate with wild female populations. Unviable eggs are produced and populations subsequently fall as a result. Other commonly used control strategies are bait sprays and trapping of male flies to reduce breeding, as well as removal of unharvested and fallen fruit that can facilitate breeding (Broughton 2019 and Prevent Fruit Fly 2019).

Trapping of fruit flies is used as an integral part of Integrated Pest Management (IPM) strategy, as an indicator for implementation of control measures. IPM has become one of the most effective pest management strategies, due to environmental concerns and increasing costs associated with traditional pesticide spraying methods (Wen & Guyer 2012). Trapping facilitates a rapid response to any outbreaks, limiting the potential for them to spread and reducing the cost of control. Both male and female trapping systems are used. The male trapping system relies on the use of pheromone lures and the female trapping system relies on a protein-rich food source as a lure. Males are also attracted to female traps (Prevent Fruit Fly 2019).

3.2 Existing insect detection systems

Currently several manual detection systems are in use for detection of fruit flies and other pests. A search of regulatory websites did not indicate that any specific designs are advocated for regulatory purposes, although specific traps are generally used for specific pests. A

selection of available trap designs that were found through web searches are shown in Table 3. For fruit fly detection, Lynfield traps are considered to be the standard method (BioTrap 2020) and for moths delta traps are considered to be standard (BioTrap 2020).

A selection of automated electronic trapping systems is summarised in Table 4. Such systems are increasingly popular to reduce costs and labour, as well as to provide more rapid information that increases the likelihood and reduces the cost of an effective response.

Table 3 Manual insect detection systems.

Description	Image
The Lynfield Trap is used as standard monitoring device for fruit flies. It consists of a disposable clear plastic container measuring 115 mm deep and 100 mm diameter at the base. It has a screw top with diameter of 90 mm and four equally-spaced holes of 25 mm diameter, 45 mm below the lip of the container. Four equally spaced drainage holes of 2mm diameter are drilled on the outer part of the base. The container lid includes a fixture to attach pheromone / insecticide wicks on the inside and a hook for hanging it on the outside. It is suspended from a branch on a wire about 15 cm long.	 (BioTrap 2020)
Bugs for Bugs Fruit Fly Traps are used for monitoring the population of fruit fly and are used with a protein attractant, combined with insecticide, that is effective at attracting and killing both Qfly and Medfly adults. They are also recommended to be used with male traps that use pheromone lures and insecticide to attract and kill male fruit flies.	 (Bugs for Bugs n.d.)

The Searles' fruit fly trap used to attract and kill male Qfly. It is used with a pheromone and insecticide lure, which is replaced every three months.

(Searles Garden Products 2017)

The eco-lure is also a fruit fly trap used to attract and kill male Qfly. It is also used with a pheromone and insecticide lure, which is replaced every three months.

(Organic Crop Protectants 2019)

The Gepro fruit fly trap is used with a Mediterranean fruit fly bait to attract and trap adult Medfly. It does not use insecticide. A similar trap is sold by the same company for trapping male Qfly using pheromone lures.

(Gepro Distributors 2016)

The Biotrap V2 X fruit fly trap is designed for use with various baits and attractants, to attract and kill fruit flies. It is advertised as lasting more than 5 years. Biotrap sells various attractants, including protein attractants for male and female flies, as well as pheromone attractants for both male Qfly and male Medfly.	 (Biotrap 2020)
The Biotrap Jackson Trap is advertised as an effective trap for Medfly. It uses a sticky insert to trap fruit flies.	 (Biotrap 2020)

Table 4 Electronic insect detection systems.

Description	**Image**
A smart trap developed at the University of Southern Queensland to trap and photograph moth pests. Photographs are then uploaded to a website for viewing and identification.	 (McIntyre 2016)

Description	Image
The Trapview system has been in development since 2013 and relies on a sticky surface to trap pests, which are attracted with pheromone lures. The trapped insects are photographed and the photographs subsequently uploaded to the cloud over mobile internet. Pests are automatically detected and counted using web-based software.	 (Trapview 2013)
A self-cleaning Trapview model, as installed for trials by the local distributor ADAMA, through the Northern basin of the Murray-Darling catchment **(Calver 2019)**.	 (Calver 2019)
The Snaptrap is an Australian invention and records high resolution images, along with temperature and humidity data for life-cycle prediction. It connects to the mobile phone network to upload images. It can be fitted onto various trap types, including Lynfield traps for fruit fly, Delta traps for codling moth, as well as other types. (Snaptrap 2017, von Hörchner 2018 and Prevent fruit fly 2017)	 (Prevent fruit fly 2017)

Description	Image
Cloud-connected fly traps with high resolution cameras have been deployed in the Goulborn Valley. Manually captured photos and GPS information are uploaded to Microsoft Azure for analysis and crowd-sourced information is distributed to participating growers. This system may be equivalent to the early stages of the snaptrap concept, although it is not clear from the source article. At the time of writing it was under continued development.	 (Gutierrez 2017)
DTN Smart Trap uses pheromone lures and standard sticky cards to trap insects and takes photos with a connected camera. Images are uploaded to the internet over the mobile phone network to be analysed using machine vision. Reports are then produced for download. Monitoring is advertised to last for an entire season.	 (DTN 2020)
The RapidAIM system was designed by former CSIRO employees and is currently being marketed as a low-cost option for automated fruit fly detection. It is suitable for both Qfly and Medfly. The technology works using capacitive sensors that detect insect movements and uploads results to the cloud using Narrow Band IoT transmissions (RapidAIM 2018, Duckett 2018, Schellhorn 2020 and Russell 2020).	 (RapidAIM 2018)

3.3 Electronic Insect Detection Systems

Automated insect detection has several potential benefits. A shortage of trained insect taxonomists makes routine identification tasks unproductive and few are specialised in specific species of interest. Additionally, manual data collection is expensive and inefficient and causes

delays in data collection where manual counting is required (Hernández-Serna & Jiménez-Segura 2014, Valan et al. 2019 and Chulu et al. 2019).

Automated insect identification research has been attempted at least since the mid-1990s, using image processing. Until the past few years all the models relied on 'handcrafted' feature extraction, using either a manual procedure or an algorithm (Valan et al. 2019). Automated identification has met with limited success until recently, due to issues such as differing morphology through different life stages, sexual dimorphism, colour morphs, differences in pose and missing limbs (Valan et al. 2019). Distinguishing features that have been used for taxonomic identification include wing venation patterns (Arbuckle et al. 2001) or outlines of wings or insect bodies as summarised by Valan et al. (2019). As early as 1990, automated fish identification was being achieved with success rates of up to 90%, using image analysis coupled with discriminant analysis (Strachan & Nesvadba 1990). Between the 1990s and about 2014, various systems using multi-criteria image processing combined with traditional machine learning classification techniques were used to develop automated insect detection algorithms. Examples of machine learning classification methods include artificial neural network (ANN) classification (Hernández-Serna & Jiménez-Segura 2014 and Miranda, Gerardo & Tanguilig 2014) and Support Vector Machine (SVM) methods (Yang et al. 2015 and Wang, Lin & Liang 2012). Accuracies varied but values of more than 98% could be achieved in some cases with accuracies in the mid 70% to mid 80% range being more common (Wang, Lin & Liang 2012 and Wen & Guyer 2012). Image processing, using global and local features, along with five different classifiers were used by Wen & Guyer (2012) with limited success, including: Minimum least square linear classifier (MLSLC), normal densities based linear classifier (NDLC), K nearest neighbour classifier (KNNC), nearest mean classifier (NMC) and a decision tree (DT).

In recent years research on species recognition has shifted towards deep learning techniques, primarily based on convolutional neural network (CNN) methods. This shift has been driven by increasingly sophisticated image recognition models developed by various researchers, with error rates below 10% being achievable since 2014 (on ImageNet data set) and below 5% since 2015 on the same dataset. Similarly the same error rates were achieved on the CIFAR10 dataset in 2016 (Khan et al. 2020). The CIFAR10 dataset contains 10 object classes. Since then the CIFAR100 dataset is has become the new benchmark. The CIFAR100 dataset contains 20 superclasses and 5 classes for each superclass (Krizhevsky, Nair & Hinton 2015). The history of predictive accuracy based on the CIFAR100 dataset is shown in Figure 2, showing that accuracies above 90% have only been achieved recently (Paperswithcode 2020).

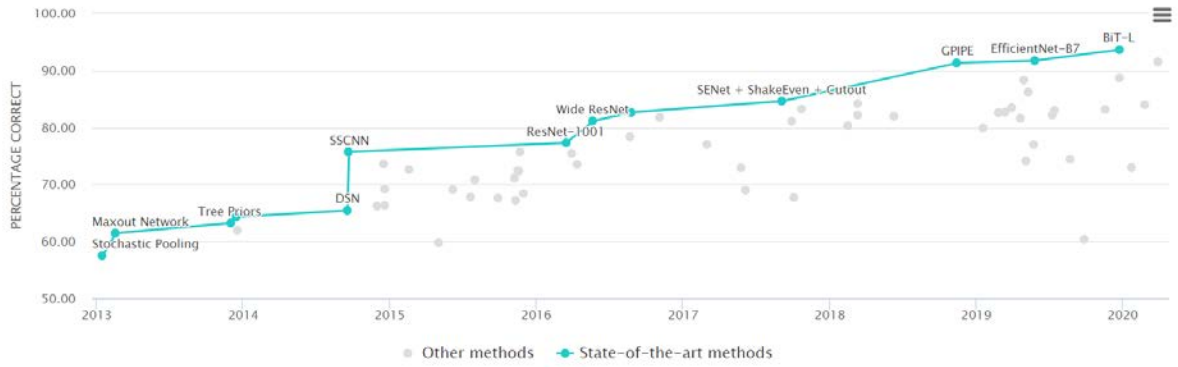


Figure 2 History of image classification accuracy for the CIFAR100 dataset (Paperswithcode 2020).

All the recent research reviewed for this work relied on deep learning methods that extract relevant classification information automatically, instead of using traditional 'handcrafted' image processing techniques (Valan et al. 2019). The basis for most of these models is various implementations and adaptations of CNNs (Glick & Miller 2016, Cheng et al. 2017, Xia et al. 2018, Sun et al. 2018, Motta et al. 2019, Chulu et al. 2019, Thenmozhi & Srinivasulu Reddy 2019 and Hansen et al. 2020). A CNN optimised by 'deep residual learning' was used by Cheng et al. (2017) to achieve an accuracy of >98% on 10 classes of agricultural pests. Valan et al. (2019) applied the VGG16 CNN model (Simonyan & Zisserman 2015) and extracted feature matrices from different convolutional layer blocks of differing dimensions, prior to MaxPooling filter layers that reduce the matrix dimensions. The extracted feature matrices were fed to an SVM for classification. The data and model for the publication by Valan et al. (2019) have been made available online.

The images that have been used for classification differ for the various CNN models that have been implemented. Valan et al. (2019) tested square images of 128, 224,320, 416 and 512 pixels and found that images of 416 pixels performed best in most models, with the exception of one that performed better at 320 pixels. The default image size for the VGG16 model is 224 pixels. Image dimensions of 450 x 750 pixels were used by Xia et al. (2018), various dimensions were used by Motta et al. (2019) and larger images were resized to 600 x 600 pixels (Sun et al. 2018) and 227 x 227 pixels (Thenmozhi & Srinivasulu Reddy 2019).

Training of deep learning CNN models requires major computational resources and a large data set. Due to this constraint it has become common practice to use a technique referred to as 'transfer learning', where a model trained on a generic dataset is used to solve more specialised tasks. This method was used commonly in the reviewed research on automated insect identification using CNNs (Valan et al. 2019, Chulu et al. 2019 and Thenmozhi & Srinivasulu Reddy 2019).

3.4 Software

Deep learning generally refers to the use of deep neural networks, i.e. neural networks with more than 3 hidden layers and up to 150 (MathWorks 2020). For computer vision the CNN architecture is generally used to detect and classify images. Major differences between what is referred to machine learning vs deep learning are that machine learning requires manual feature extraction and deep learning does not and the algorithm converges with machine learning but scales with the data with deep learning (MathWorks 2020).

Currently the Python programming language is favoured for the development of deep learning applications. Python is relatively easy to learn and has many free software libraries available for data processing and analysis. Two deep learning packages that are currently favoured and that are being actively developed are Tensorflow 2.0 (developed by Google) and PyTorch (originated at Facebook) (Stöffelbauer 2020). Both packages are capable of using GPU acceleration on Nvidia graphics cards running Nvidia CUDA software, which is a parallel computing platform that enables general computing on graphics processing units (Nvidia n.d.). Tensorflow has inbuilt GPU support, whereas PyTorch can interface directly with Nvidia CUDA support (Kurama 2020).

For computer vision purposes the current preferred software is OpenCV 4, which is freely available and actively developed. It is available with C++, Python, Java and MATLAB interfaces and Windows, Linux, Android and MacOS installers. OpenCV 4 was released in November 2018, with the current version being Ver. 4.4.0., released in July 2020 (OpenCV 2020).

3.5 Hardware

The Raspberry Pi 4B and Nvidia Jetson Nano are currently the dominant low-cost development platforms for machine learning applications. The Jetson Nano was much more expensive at project commencement but the prices have since converged, with the the Raspberry Pi 4 (4GB version). The Jetson Nano is a much faster platform for use in machine learning and machine vision applications. This is due to an on-board CUDA enabled GPU processor. October 2020 pricing for the Raspberry Pi 4B (4GB) is AU$130 and for the Jetson Nano AU$ 200.65 from Core Electronics. Several other platforms are available but are generally more expensive (Q-engineering 2020). For applications that do not require real-time processing, slower computers are adequate. As real time processing is not envisaged for the proposed application, consideration was also given to Arduino platforms and the Raspberry Pi Zero.

A literature search of software capable of running machine learning models on microprocessors indicated that Tensorflow Lite for Microcontrollers (e.g. Arduino) is currently in development (Tensorflow 2020) but will only be able to run with limited functionality. All Raspberry Pi models are able to run Tensorflow Lite or the full Tensorflow package (Duncan 2019), although memory size and processing speed will limit which programs can be run on particular models of Raspberry Pi. PyTorch can be run on Raspberry Pi 3B and 4B models (Kanda 2018 and Das 2020) and PyTorch Mobile is in development, with likely future support for Raspberry Pi (Johnson 2019).

In addition to software considerations, the computing options for high resolution cameras were also investigated. Low resolution cameras are not considered suitable for identifying multiple insects from single photos, although could be considered if insects could be positioned within a limited frame. For single image captures low resolution images will not provide adequate resolution for reliable image classification. Low power microcontrollers such as Arduino are incapable of running high resolution cameras, thus making Raspberry Pi computers the most affordable feasible option. Many high-resolution camera models are available for use with the Raspberry Pi, with the highest resolution that could be found through an internet search being 18MP, although a 16MP option was more likely to be a practical solution due to a substantial cost difference between the two models (Arducam 2019).

To upload data from field units, various communication options are available. Most remote systems on the market rely on 3G and 4G mobile networks to operate (Trapview 2013), (Snaptrap 2017). In areas where poor mobile access is available, such as various parts of rural Australia, alternative methods must be considered. A review of options available for Raspberry Pi computers found that in addition to traditional mobile internet, three other viable options may prove more useful in rural areas, namely Narrow Band IoT, LoRa and long distance WiFi.

Narrow Band IoT is a radio transmission technology supplied by mobile internet service providers and is the technology used by the RapidAIM system (Russell 2020). It operates in a narrow band of the LTE spectrum, specifically for interfacing with IoT devices. It is installed along other mobile technologies at cellular base stations. Although this technology is slower than broadband mobile internet, it requires less energy and provides much greater areal coverage. Uplink data rates of 127 kbit/s and download rates of 159 kbit/s are achievable for the LTE Cat NB2 standard (Lupori 2020).

LoRa (Long Range) is a radio transmission technology used for data transfer. It can transmit data for up to 30 miles (48 km), has low cost and power requirements and accommodates encryption and geolocation (Semtech n.d.). It is limited to intermittent transfers at low data

transfer rates (250 bit/s to 11 kbit/s), so can only be used for low-volume data reporting purposes rather than large-scale data transfers (The Things Network 2020).

Long range WiFi uses standard WiFi protocols but achieves greater transmission range than domestic and commercial WiFi systems through application of higher power and / or directional antennas. Broadband transmissions at ranges of 30+ km are achievable (Ubiquiti 2020). Cost and complexity can be expected to be greater for long range WiFi than for the aforementioned systems.

In many locations standard mobile broadband services may be usable and this functionality can easily be included as a data transfer option along with any of the others. Raspberry Pi hats[1] are available to implement all of these technologies.

4 Aims and Objectives

The aim of this research was to develop an insect identification and counting system capable of cost-effective monitoring of insect pests. The system would be built as cheaply as possible, using low-cost hardware and open source software. Effective data processing and a communications protocol that would enable the system to operate as a network were additional aims.

The research was to be done on Medfly in the Perth Metropolitan area of Western Australia. Hence the monitoring system would be specifically aimed at complying with the specifications of the Lynfield trap, which is the standard monitoring system for this species presently. Consideration would also be given to designing the system to be adaptable for other species. This would help to improve species discrimination and would simultaneously be useful for expanding the use of the apparatus for other species. Identifying other species was not a specific objective of the project.

The following objectives were set at the start of the project:

- Collect fruit flies in a trap or traps to enable the collection of photographic data.
- Capture photographs of trapped fruit flies that are representative of photographs that can be captured by a Raspberry Pi Camera.
- Set up a Raspberry Pi computer to capture images from a camera and readings from a pressure, temperature, humidity sensor. Potentially include a GPS module for automated position reporting.

[1] Add-on circuit boards with specific additional functionality.-

- Install suitable software to process all data and to potentially transmit it to a receiving computer.
- Obtain additional images for model training and calibration from other sources.
- Utilise OpenCV software for insect image extraction and pre-processing and Tensorflow software for image classification, using the method of Valan et al. (2019). Process all data on-board the Raspberry Pi computer to remove the need to subscribe to online services or transmit images unnecessarily, so that low bandwidth data communications can be used and additional computer equipment will not be needed.
- If time and resources permit, optimise hardware specifications based on performance characteristics of the final model, to minimise the cost for a viable monitoring system.
- If time permits, establish and implement a method for sending relevant data by WiFi, LoRa or Narrow Band internet methods, for application in areas with poor mobile internet coverage.

5 Materials and Methods

5.1 Study site

Since Medfly is prevalent throughout the Perth region and Western Australia in general, a study site was to be established at home to test the prototype equipment in a practical context. In recent years there have been signs of fruit fly damage, suggesting this study site would be effective. Being able to work at home facilitated the use of home wifi to test communication and enabled manual checking and modifications to equipment during development. It also avoided unnecessary travel during the COVID-19 pandemic, with possible ‘lock-downs’ being an ongoing concern throughout the project period. A location map of for the study site is shown in Figure 3.

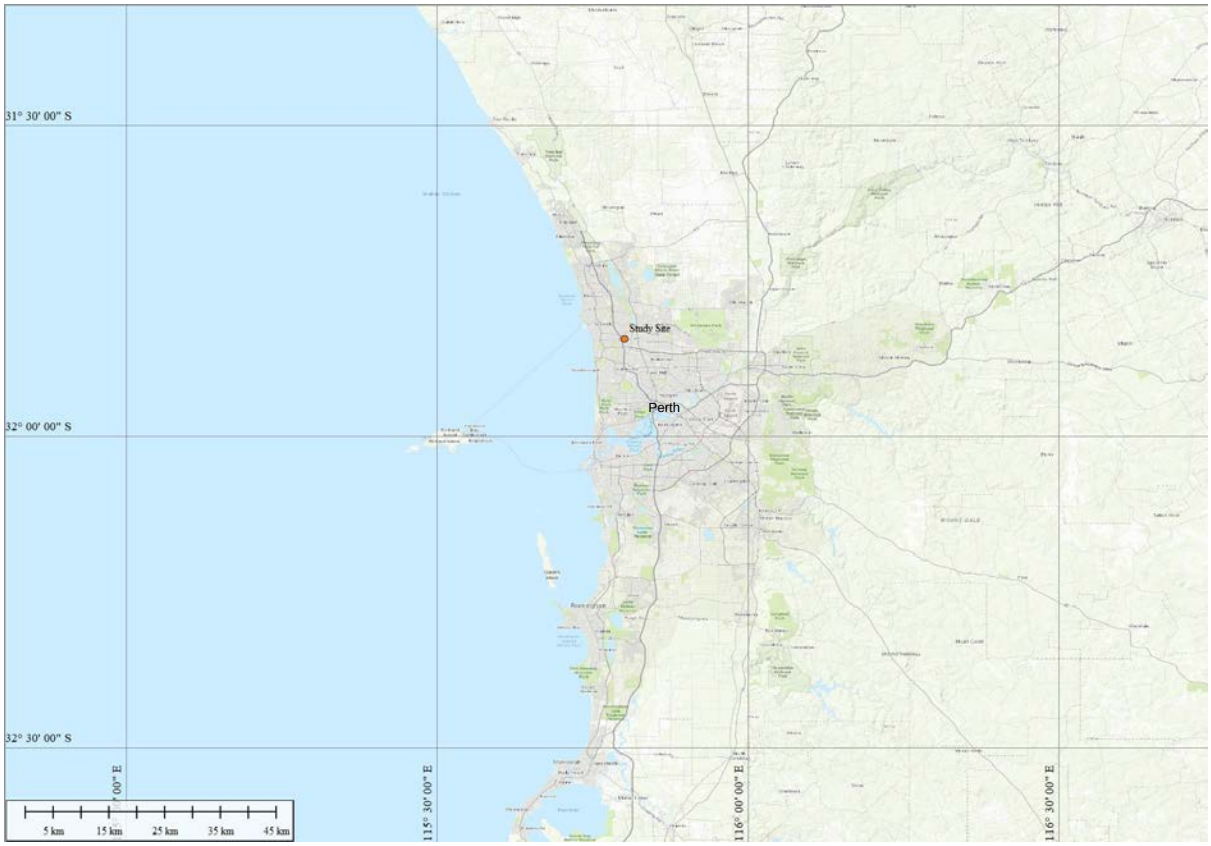


Figure 3 Location map for study site.

5.2 Computer hardware and software

The following computer hardware and software, as well as consumables were specified for the project:

- Raspberry Pi Zero W and Raspberry Pi 4B (8GB) model.
- Bosch BME280 temperature, pressure and relative humidity (PTH) sensor to record weather conditions affecting the Medfly breeding cycle.
- U.USV Professional power management hat, to power the Raspberry Pi on and off at defined intervals, to conserve batteries (only to be implemented if adequate progress was made on other tasks). This hat is also capable of managing battery charging if solar charging is to be implemented in the field.
- Lynfield trap (by BioTrap).
- TriMedLure Pheromone attractant wicks and DDVP (Dichlorvos) insecticide cubes (by BioTrap).
- Arducam IMX298 16MP camera. Calculations based on the 100 mm diameter of the Lynfield trap showed that a 3 mm fruit fly will occupy about 105 pixels along its length, for the minimum dimension of a 16 MP sensor resolution (3496 pixels). The highest resolution official Raspberry Pi camera at this time is the 12.3 MP version, which would

achieve a section of 81 pixels over a 3 mm minimum length of a Medfly, thereby substantially limiting visible detail compared with higher resolution options. An 18 MP Arducam camera is available for the Raspberry Pi but is too costly to justify given the project objectives and budget (US$ 120 vs US$60 for the 16 MP camera), although it would improve resolution to 118 pixels over a 3 mm length. The 16 MP camera was also found to be suitable insofar that it has a 60° field of view. The presence of an electronic lens adjustment option was also expected to allow for implementation of autofocus, which would allow for sharp images down to a minimum object distance of 5 cm. This would allow for most of the field of view to be filled by the base of the trap if the design could be optimised, assuming an optimal camera housing setup.

- Free software, including:
    - Raspbian (renamed Raspberry Pi OS) operating system.
    - Python with SciKit-Learn, OpenCV 4.4.0 and TensorFlow 2.0 libraries.
- A trap housing was designed in Autodesk Fusion 360 and the parts were 3D printed at USQ and posted to me.
- An MB3725 multi-port USB portable charger was used for the power supply.

5.3 Hardware design

The hardware design was constructed as planned, based on the equipment specified in Table 5 but excluding the S.USV energy and time management hat, as time did not permit for the time and power management functions to be implemented.

The Raspberry Pi 4B (8GB) computer was set up according to instructions provided according to (Raspberry Pi 2020) and (Core Electronics 2019). Both headless operation and GUI operation are required, as headless operation allows for remote control, which is necessary for controlling the apparatus inside its housing and GUI operation allows for working directly on the Raspberry Pi computer and viewing outputs. Headless operation and file transfers were done using the secure shell (SSH) communication protocol via PuTTY connection software and FileZilla software respectively. A Raspberry Pi Zero W computer was also set up in the same way and was found to be capable of running the Arducam camera but with slow overall performance. This was considered to be a major drawback for development purposes but may be acceptable for field implementation if the computer can run the Tensorflow code required for fruit fly identification. The frequency of monitoring and computing requirements would also be a consideration if long computing times are required.

The Arducam IMX298 16 MP camera was set up according to the instructions provided in (Arducam 2020 and GitHub 2020). After setup it was discovered that the camera resulted in an error when running on the Raspberry Pi 4B 8GB, which was found to be due to the 8GB

version of the Raspberry Pi 4B not yet being supported by the MIPI camera driver software. Arducam provided a software update on 28th June 2020, which enabled the Arducam MIPI software to be used after it was reinstalled. Due to slow performance development on the Raspberry Pi Zero W was stopped at this stage, with a view to re-evaluating it once the required computing resources were known.

Once operational, the different algorithms in the MIPI software suite were tested in an attempt to obtain good quality images from the IMX298 camera. This phase of the work became a major bottleneck in the project, as the camera software and camera hardware do not have a manual and are relatively poorly documented in terms of their capabilities. The demo algorithms provided were modified in an attempt to activate autofocus, to change timing to allow for adjustments to stabilise and to adjust white balance and exposure. Although moderate success was achieved for some tests, it was revealed by Arducam technical support on 17 September 2020 that the camera did not officially support automated functions due to the lack ISP functionality on the camera. Further development of the camera implementation was abandoned.

The Adafruit neopixel 2852 was set up according to (Rembor 2020), using the level shifting chip option. The power supply is through a USB3 connector, allowing it to be run from a separate connection to the same power pack that powers the Raspberry Pi computer.

The Bosch BME280 PTH sensor was installed according to (Matt 2016) and suitable start code, with an MIT license, was obtained from (Nicolai 2020) to read sensor data.

The 3D design that was used to create the 3D printed housing is shown in Figure 4, with the Raspberry Pi computer and the intended S.USV power / clock board indicated (downloaded from grabcad.com). The assembled electronic components are shown in Figure 5. The base of the assembled apparatus is shown in Figure 6 (excluding the trap base) and a view with the neopixel illumination activated, with electronics exposed is shown in Figure 7. The fully assembled and deployed device is shown in Figure 8.

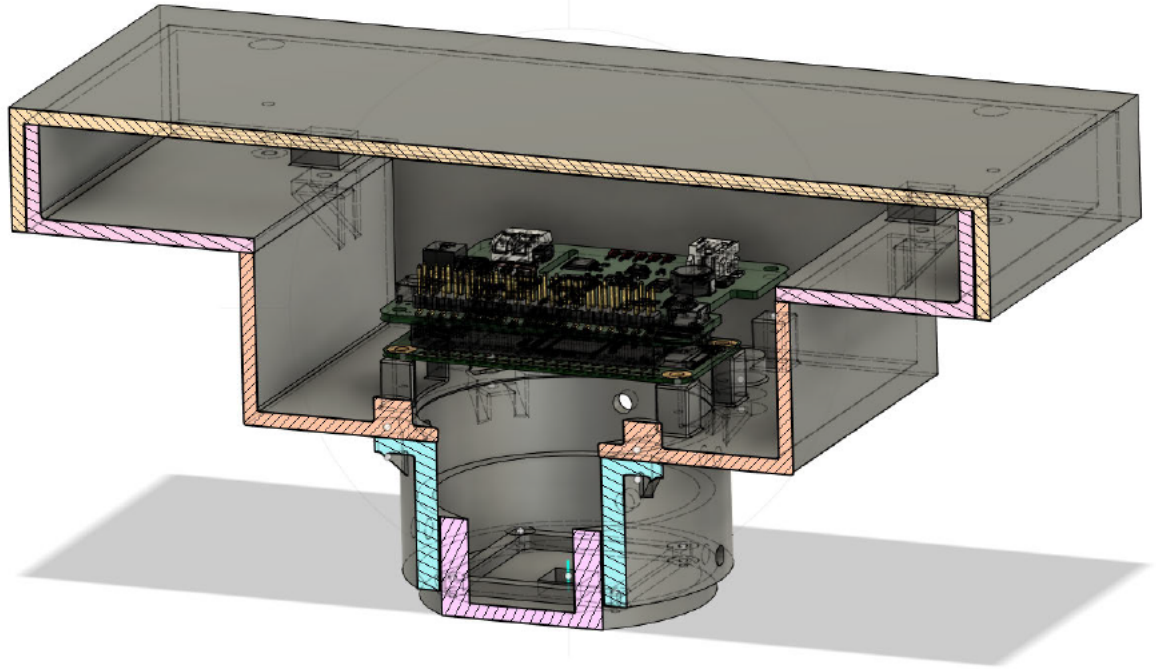


Figure 4 3D design of housing with Raspberry Pi computer and S.USV power / clock boards indicated.

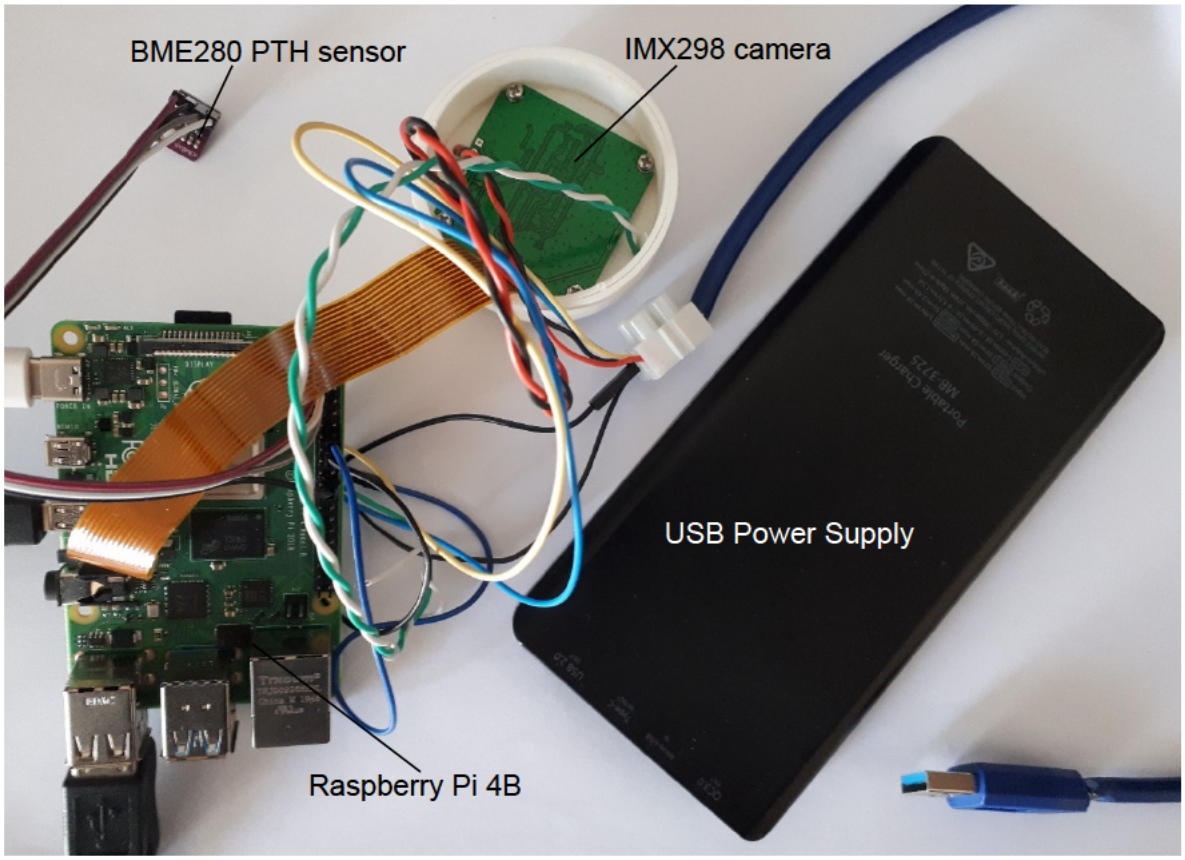


Figure 5 Electronic components, including all connections except USB power.

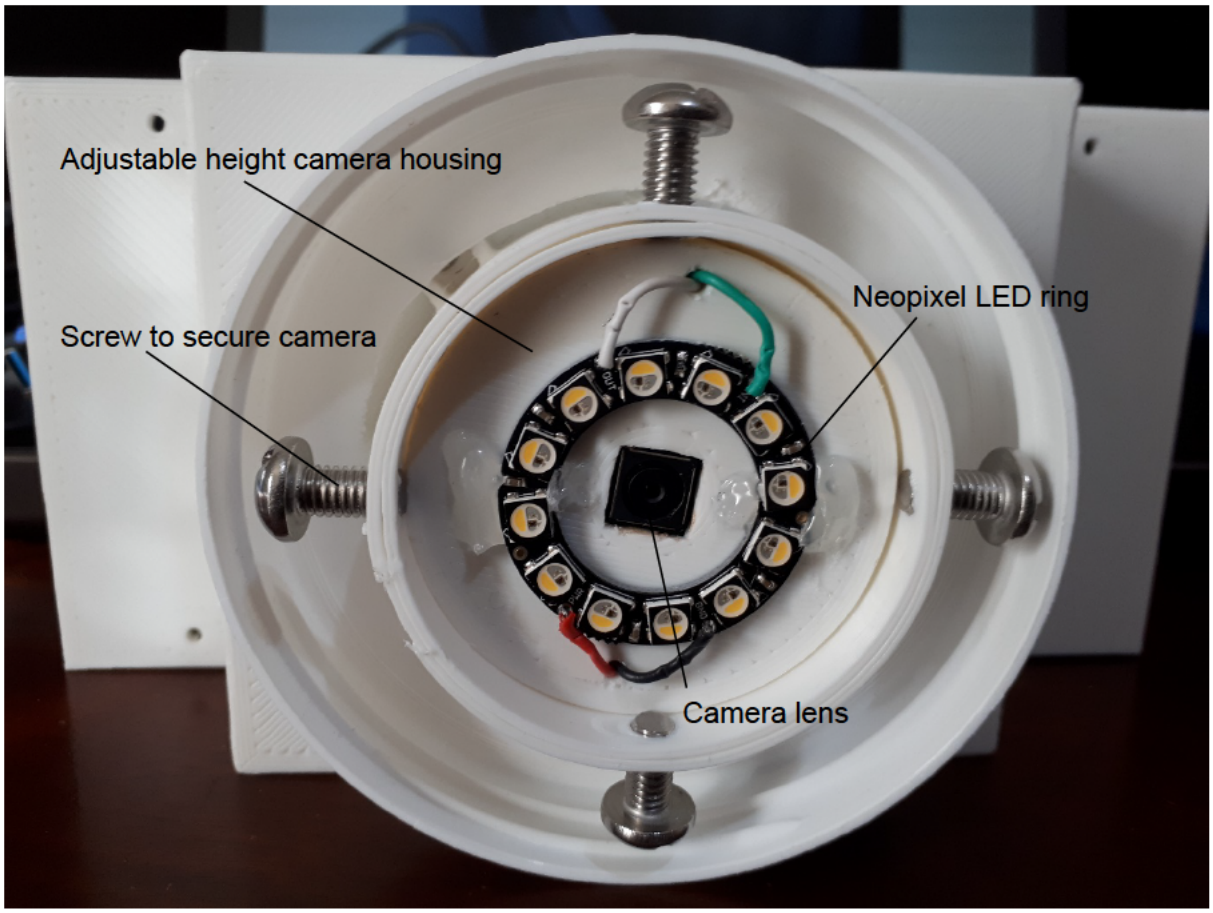


Figure 6 Base of apparatus showing neopixel illumination, camera lens port and adjustable camera housing.

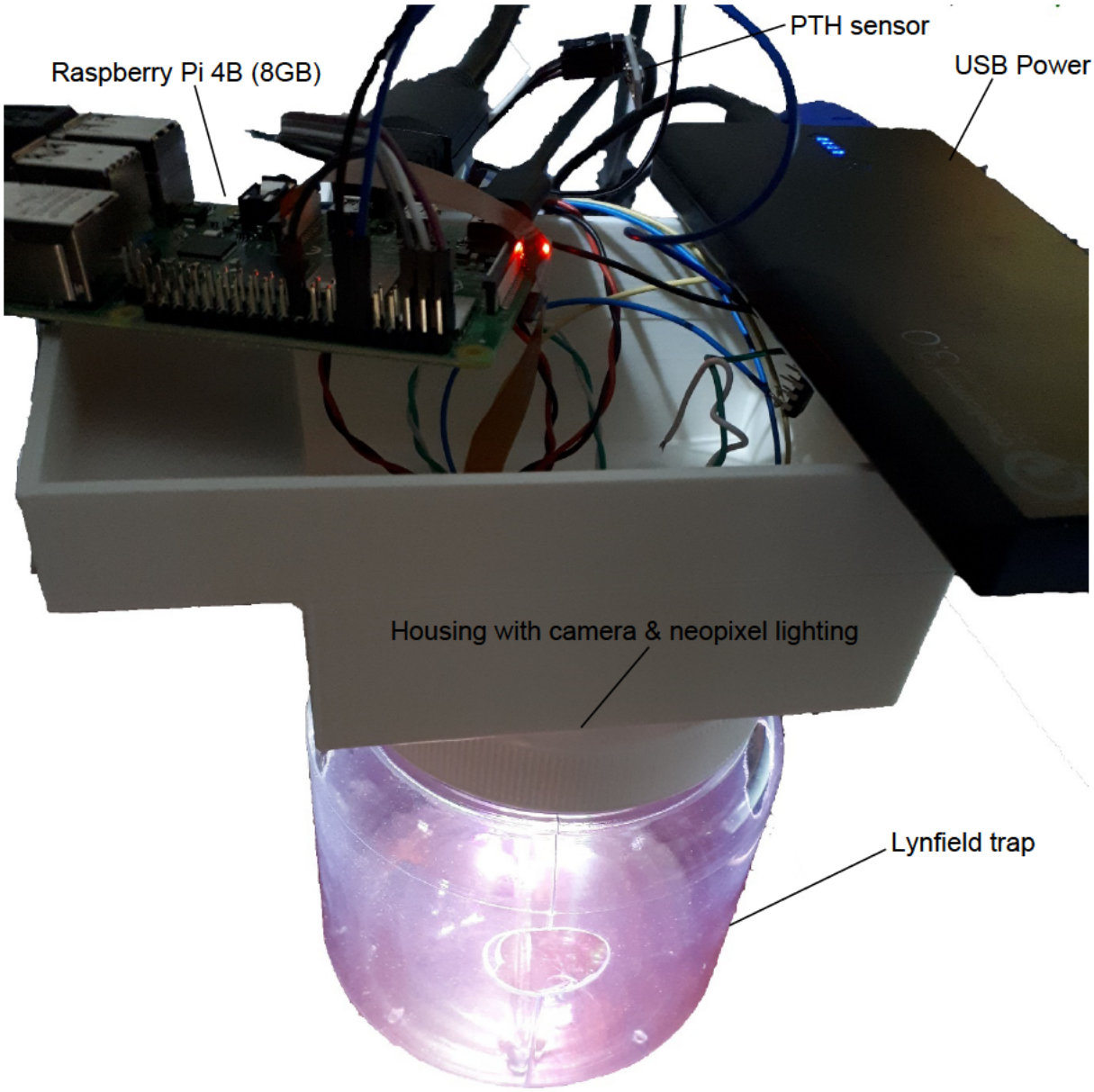


Figure 7 Assembled apparatus with exposed electronics and with illumination active.

Figure 8 Photo of fully assembled trap

5.4 Software

5.4.1 Image collection and processing

The software design flow chart is shown in Figure 9. This approach allows for parallel processing so that a camera image can be obtained while simultaneously operating the neopixel illumination. Without parallel operation the camera algorithm would have to wait for the neopixel algorithm to finish running before a photo is taken, which would prevent correct operation. Both active and passive operation of the neopixel can be used for simultaneous operation. For passive operation a call can be made to start the neopixel within the camera control software and later a separate call can be made to stop it, after the image has been

captured. This is possible because the neopixel can be set to retain its state until it receives a signal to change it. For active operation a conventional parallel processing approach is required, which uses a subprocess to run the neopixel separately but under the control of the main algorithm.

Parallel or prior operation of the PTH sensor is also considered desirable because of heating of the CPU that will occur during operation of the Raspberry Pi computer, thereby likely affecting measurement accuracy after a period of time, as the heat is dissipated through the housing and potentially the surrounding air. The Bosch BME280 sensor is also prone to self-heating (Adafruit forum 2016) during operation, which implies that measurements should be taken as soon as possible after the sensor is initialised to minimise this effect. If PTH measurements are to be collected on a different schedule to images, parallel processing and separate time management processes would be required. To maintain data integrity and allow for periodic data transmission, that is not necessarily real-time, PTH data is stored in a csv file for later retrieval.

The Arducam IMX298 image is saved as a jpeg file for processing and optional retrieval. The saved jpeg image is then retrieved by an OpenCV algorithm and processed for fruit fly detection and identification. Once identified fruit flies have been counted the count data is collated with PTH data for the relevant period into a new csv file and sent to a designated email address. Once transmission is complete the Raspberry Pi is shut down. The S.USV power and time management board would control start-up and shutdown routines in a fully implemented system, however, due to various complications and delays, as well as the manual for the S.USV board only being available in German, the power and time management functions could not be implemented in the available time. These operations were therefore still done manually at the stage of the project that was reached.

Implemented python code is included in Appendix B.

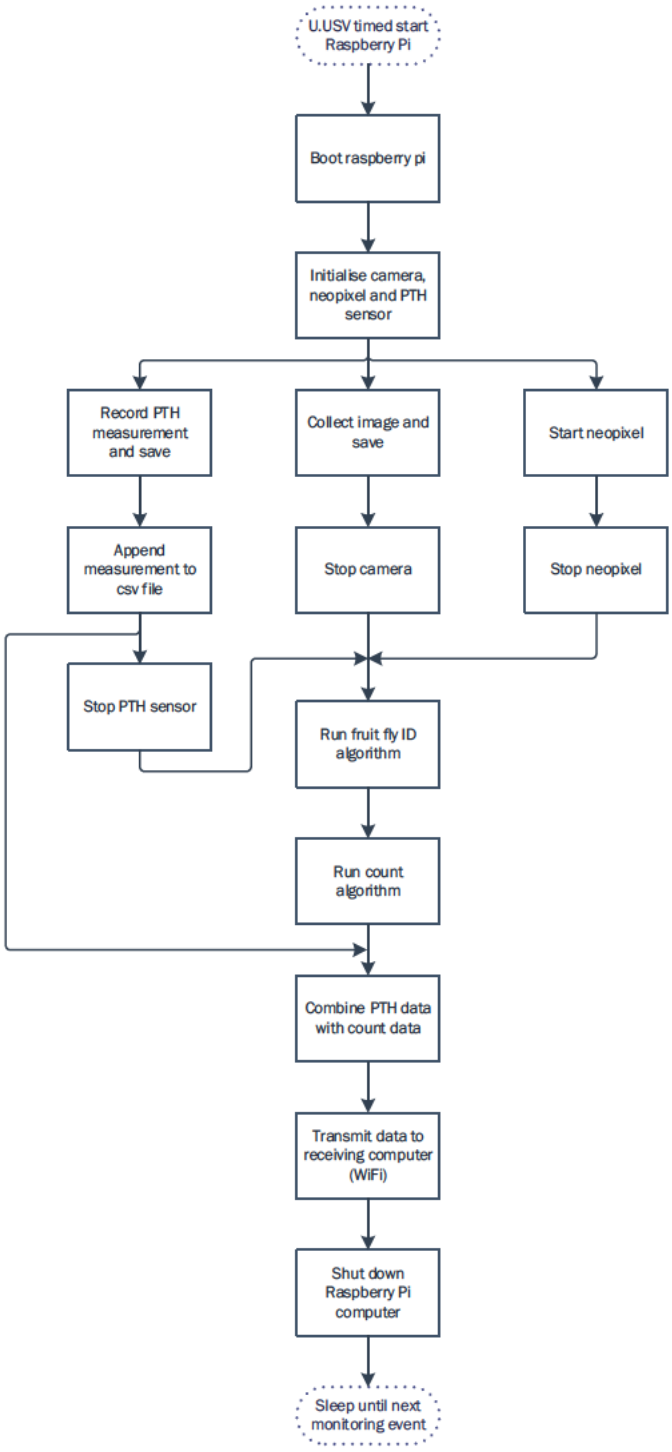


Figure 9 Software design flow chart.

5.5 Medfly detection & identification

The intended aim of the project was to create an insect identification system capable of discriminating Medfly from other species of fruit flies, with the added benefit of being able to detect other species as well where they are present, depending on the training images used. It was intended to use the algorithm published by Valan et al. (2019) to achieve this, as it had been demonstrated to have high accuracy and was programmed in python.

The process of implementing this system was hampered by several issues, which are summarised below:

- Due to the difficulty in obtaining good quality images with the Arducam IMX298, discrimination was expected to be poor until this could be rectified.
- The algorithm of Valan et al. (2019) was written using older versions of Python, OpenCV and Tensorflow, which complicated integration between different software components and required that the code be updated.
- The algorithm of Valan et al. (2019) requires additional software development to be implementable as intended. The algorithm is only designed to identify insects from single images. Hence, to be useful for identification for this project, prospective insect images first need to be extracted for processing by the algorithm or the algorithm's functionality must be extended to search for fruit flies prior to identifying them.
- The adequacy of the training dataset is doubtful and, both in terms of size and quality. Typically thousands of images are required for a good training dataset. The training dataset assembled so far contains 711 Medfly images, of which many are duplicates and dorsal views. These factors indicate that the dataset in its current form would be both insufficient and biased if used for training.
- A test run on the Raspberry Pi 4B (8GB) found that the algorithm of Valan et al. (2019) had insufficient memory to run in its original form. The code therefore needs to be modified and / or tested using the 64-bit Raspberry Pi operating system that can address the full computer system memory. The 64-bit operating system is still in the beta development stage at the time of writing and hence compatibility and reliability is expected to be a concern. The high memory requirement also indicates that running this algorithm on the Raspberry Pi Zero W will be challenging, if at all possible. Conversion to a Tensorflow lite version would be a minimum requirement.

Based on these considerations, a simpler system was sought for initial detection and potential extraction of image frames for identification. After a process of elimination the OpenCV Simple blob detector method was investigated further, the results of which are discussed further in section 6.4.

More sophisticated algorithms were considered but were eliminated due to requiring relatively accurate image matches relative to the variability in lighting, pose and orientation displayed by fruit flies in a trap. To accommodate such variability only the deep learning approach is expected to work satisfactorily (Goodfellow, Bengio & Courville 2016 and Valan et al. 2019). This was also the intended method to be used for the project but could not be implemented due to the reasons mentioned above.

5.5.1 Training images

Training images were obtained from the internet. Images for the target species of this study were sought from idigbio.org (the same source used by Valan et al. 2019) but were generally found to be of old museum specimens that would unlikely be representative of recently caught fruit flies, especially in terms of colour, since colours are generally faded. Many of the photos were also of low quality. The use of Google image search was limited due to downloaded images being limted to a number of 300 and available tools for downloading them being poor. Consequently images were downloaded using the *Download All Images* extension in the Firefox browser, using the DuckDuckGo search engine.

In total 21 612 images were downloaded using this option, to be subsequently manually 'culled' to 1421 images of fruit flies (711 for Ceratitis capitata, 418 for Bactrocera dorsalis, 289 for Bactrocera tryoni and 3 for Ceratitis cosyra). Due to the limited number of Ceratitis cosyra images it was decided not to use this species, despite it being relatively similar in appearance to Ceratitis capitata and thus making it desirable to include to improve the trained model's discrimination ability.

Species were verified to be the correct species as far as was possible, although it is likely that some incorrect identification remains due to strong similarities between some species. Many images were also duplicated several times, a problem which is more difficult to eliminate manually in such a large dataset and which may cause biasing in the neural network – SVM model if not removed. Images in which fruit flies were a minor component were cropped to reduce background influence The prevalence of dorsal and frontal views of live flies is also a concern, as other angles and poses of dead flies are unlikely to be sufficiently represented in the dataset.

Specimens of fruit flies were sought from the Department of Primary Industries and Regional Development of Western Australia (Agriculture and Food Division) for model testing and training purposes. In this regard, Dr. Sonya Broughton of the Department of Primary Industries and Regional Development – Western Australia (Agriculture and Food – Biosecurity) was contacted. She arranged for the supply of two separate vials of male and female Medflies for the study. This supplemental material was considered to be necessary, since low fruit fly activity was expected during the bulk of the project duration and only male Medflies were expected to be caught with pheromone lure. This was because the project was mostly completed during the winter months when fruit fly activity is lowest. It was therefore expected that representative specimens and / or photos were unlikely to be available for model training purposes solely through local trapping efforts.

Additional images were collected using these specimens as planned. An initial set of 14 photos was collected while testing the Arducam IMX298 camera, as well as 63 images using an Andonstar V160 electronic microscope for higher resolution images. Due to the time consuming nature of collecting images with the microscope, this approach was abandoned with a view to resuming it if time permitted at a later date.

The IMX 298 camera was found to produce relatively poor quality images. This is attributed largely to the unrefined hardware and software for this camera, rather than inherent limitations of the 16MP resolution. Examples of the best quality images obtained from the Andonstar V160 and the IMX 298 at this stage are included in Figure 10 and Figure 11. A zoomed section of the image, as indicated with the red box in Figure 11, is included in Figure 12 to demonstrate the image quality achieved. *Note: Figure 11 was obtained using oblique lighting during camera testing and is not representative of the results obtained with the neopixel lighting system, which resulted in more reflected light from the base of the trap.*

The adequacy of using a 16MP camera was demonstrated by collecting images with a Samsung A7 2017 mobile phone camera, which also has 16MP resolution. A much better image was obtained this way, as shown in Figure 13.

Figure 10 Medfly image taken with Andonstar V160 electronic microscope.

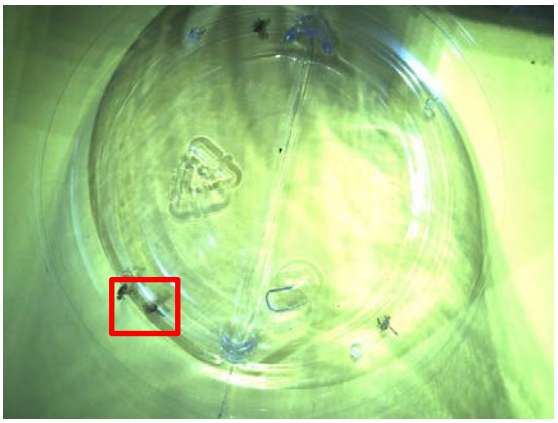


Figure 11 Image of four Medflies inside a Lynfield trap, taken with Arducam IMX 298 camera.

Figure 12 Zoomed section of IMX 298 image, as indicated with the red box in Figure 11.

Figure 13 Medfly image taken in Lynfield trap with a Samsung A7 2017 camera at 16MP.

5.5.2 Weather data collection

Python algorithms were developed to collect TPH data from the Bosch BME280 sensor, based on start code provided on GitHub (Nicolai 2016), which provided access to sensor readings. An initial algorithm was developed for short term readings, constrained only by the sensor update rate (between about 27 and 55 ms). This was used to measure noise effects on the sensor readings for a test of duration about 10 minutes. The algorithm was subsequently implemented with a delay (default 30 seconds) to be able to collect readings over longer periods without excessive datapoints. For this option 15 short-interval readings were taken in succession to smooth the noisy data stream and the average was recorded with an adjustable delay between sets of readings. This algorithm was used to collect readings with 5 second delays for about 18 hours, to evaluate drift effects. It was then set up to run for 12 days with a 30 second delay between readings, to collect real weather data, which could be used for modelling purposes. This assembly was placed below an open patio shelter to avoid equipment damage, while also attempting to sense real ambient conditions. Due to the late stage of the project when this system was developed this data is only suitable for demonstration purposes and of limited use for analysing trap data.

5.6 Communication protocol

A home WiFi system was used for communicating with the Raspberry Pi 4B over its inbuilt WiFi connection to run live demonstrations. Due to the limitations of the camera, remote photography could not be implemented. Remote collection of weather data was, however implemented successfully as discussed in section 6.2. The Raspberry Pi computer was located about 15 m from the WiFi router and remained connected, except for a few WiFi outages over the period of data collection. Remote connections to run and check the Raspberry Pi computer were made using the secure shell (SSH) protocol. Remote terminals were opened to enable interaction with the Raspberry Pi using PuTTY Ver. 0.73 and file transfers were achieved using Filezilla Ver. 3.50.0.

To avoid automatic shutdown of processes that were started through the remote terminal terminated upon exiting the PuTTY, the Linux tmux extension was installed and implemented. This allowed for the Raspberry Pi to be disconnected from the control computer and to operate independently.

Further investigation of alternative communication options was not done due to inadequate time.

6 Results and Discussion

6.1 Build costs

The total build costs for the project, excluding the 3D printed housing, are as follows:

Table 5 Costs of equipment specified for construction.

Equipment	Cost (AU$)
Raspberry Pi 4B 8GB	$ 152.95
Arducam IMX298 camera	$ 133.10
SanDisk Extreme 64GB SD card	$ 22.80
Bosch BME280 PTH sensor	$ 8.19
Adafruit neopixel model 2852	$ 16.06
74AHCT125 logic level shifter chip	$ 0.91
S.USV Advanced energy management and clock	$ 86.99
USB powerbank	$ 34.95
Lynfield Traps with TriMedLure	$ 8.74 each
DDVP insecticide cubes	$ 10.70
Cabling (USB and connector cables)	$15 approximately
TOTAL	$ 490

The costs reflected in Table 5 are considered to be the maximum cost for construction of the fruit fly trap, since cheaper alternatives are available for most components once actual resource requirements are established. Examples for possible savings include cheaper models of Raspberry Pi, depending on actual computing requirements, a lower capacity SD card, cheaper suppliers of Bosch BME280 PTH sensors and power management options other than the S.USV power board.

6.2 Weather monitoring

6.2.1 Sensor stability readings

The results for stability readings for Bosch BME280 PTH sensor, to evaluate short and medium term noise and drift, are included for a ten minute test in Figure 14. It can be seen from this test data that there is high frequency noise within the data of about 0.02 °C on average, with a maximum spike of about 0.05 °C at about 455 seconds. Self-heating of about 0.5 °C over the first 4.5 minutes of the test is also visible, with subsequent fluctuations in readings of about 0.1 °C likely to be attributable to real fluctuations in ambient temperature caused nearby work activity.

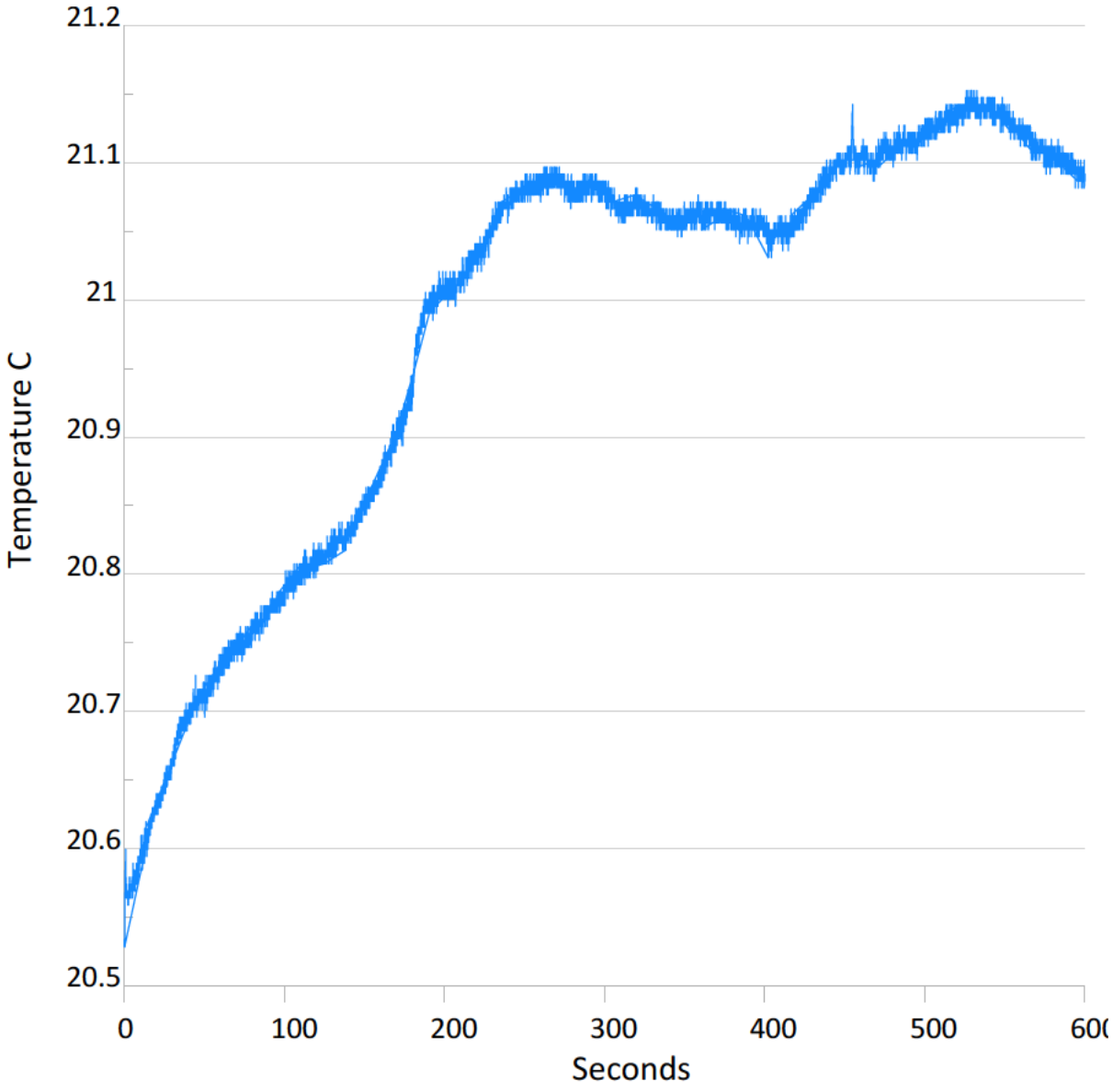


Figure 14 Plot of raw test data vs time in seconds for BME280 PTH sensor for 10 minute test.

Plots produced with Statistica 12.7 in Figure 15 to show the effect of the averaging algorithm that was implemented for data smoothing. The graphs are for raw data and averaged data over 5, 15 and 100 sampling points. Whiskers show the range of values for all the points within each interval without smoothing. It can be seen that much less erratic data is generated through recording averages of multiple readings. Although the temperature differences observed are not of practical importance, this approach is considered to be better practice than collecting single readings, which may be affected by larger deviations from time to time.

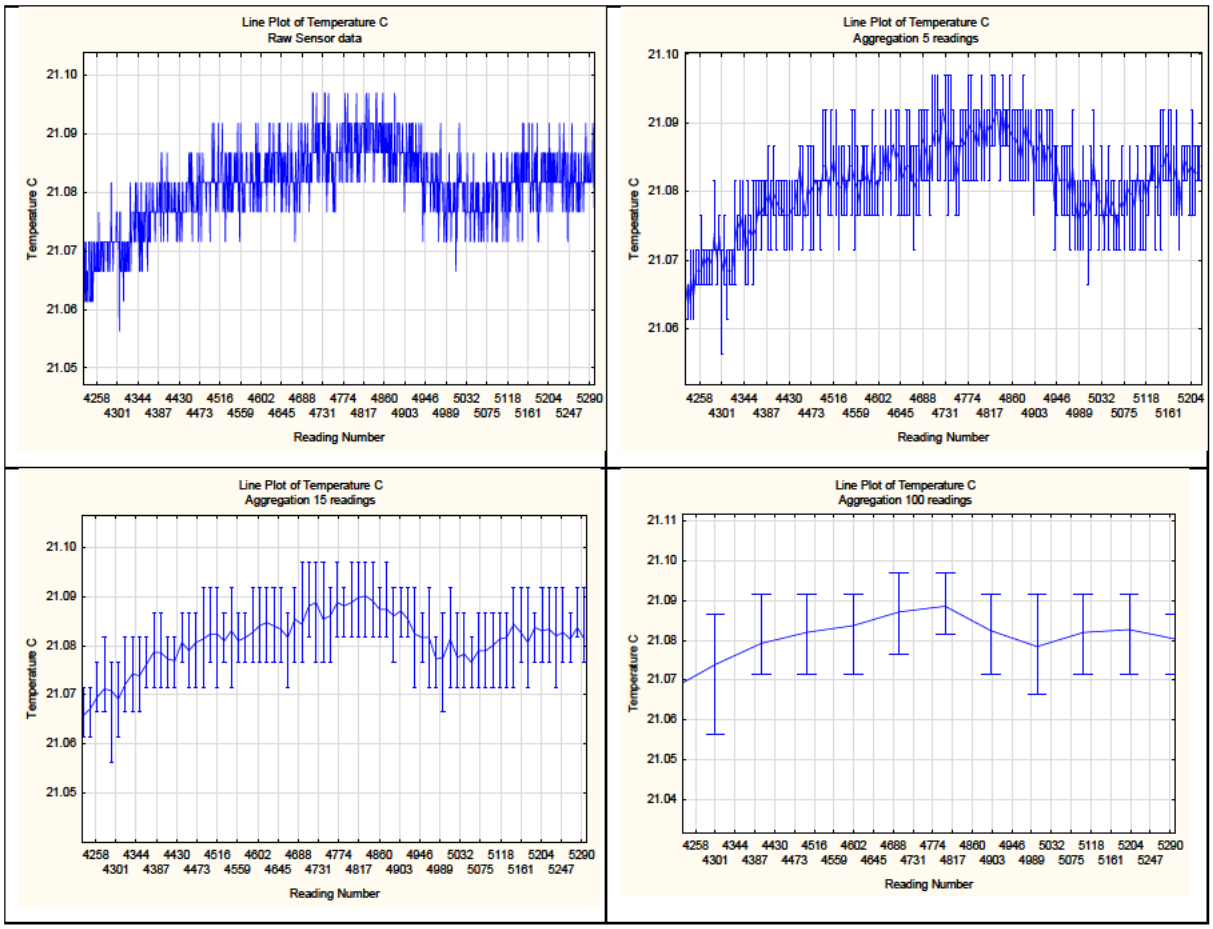


Figure 15 Series of figures showing raw data and moving average aggregation of 5, 15 and 100 points from readings 4230 to 5300, corresponding to the interval 235.41 sec to 295.05 sec.

A longer term test, over about 18 hours (Figure 16), was used to check the behaviour of the sensor prior to deployment. The data obtained shows the temperature profile that is very sensitive to activity around the sensor. This variability is representative of real conditions, as during the night, when no activity occurred, the temperature decreased smoothly, with the exception of a blip of about 0.25 °C at about 6:00, the cause of which is uncertain. The temperature increased again after activity commenced in the office and as the weather warmed during the day. Based on this data and test data published by Smith (2018) the accuracy of this sensor should be adequate for monitoring purposes, although it would be beneficial to compare it with an accurate calibration method as individual sensors are expected to vary.

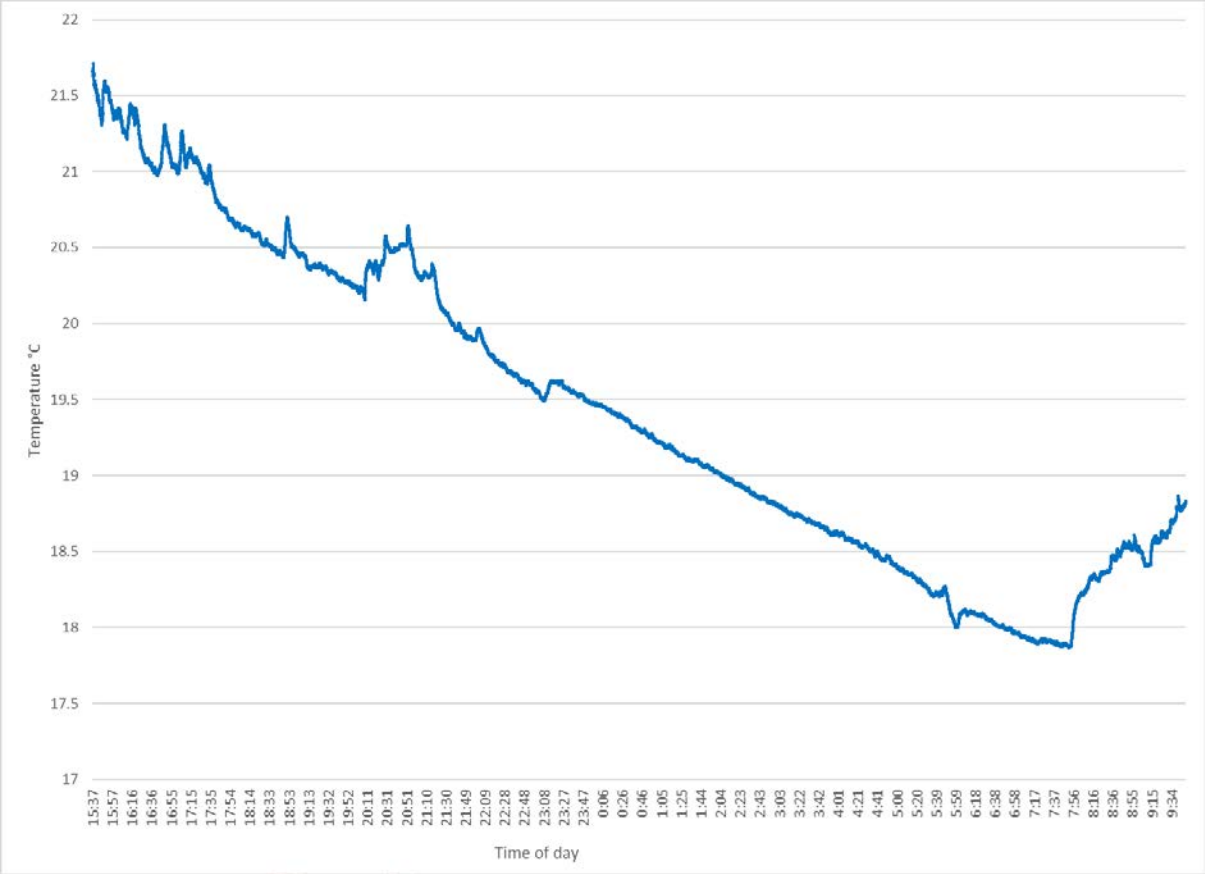


Figure 16 Eighteen-hour test run to evaluate PTH sensor stability.

6.2.2 Long term weather measurements

Longer term weather measurements could only be made between 30 September 2020 and 11 October 2020 due to the time required to construct the necessary hardware and develop the software for necessary apparatus and the requirement to process the data before the project deadline on 15 October 2020. The collected data is shown in Figure 17.

It can be seen from this graph that temperature and relative humidity measurements were more erratic during the warmer periods of the day. This is attributed to the position of the sensor likely being exposed to direct sunlight at certain positions of the sun but could also be influenced by changing microclimatic conditions. This would have to be investigated further by changing the position of the sensor and / or checking for correlation between increased sunlight and temperature peaks.

Comparison of the collected data with daily data from the Bureau of Meteorology (Bureau of Meteorology 2020) shows that the temperatures generally track each other but that the study site is generally a few degrees warmer. A few exceptions occurred where the minimum temperature was slightly colder at the study site on the 1st, 2nd and 5th of October than reported for Perth Airport. The larger difference between maximum temperatures than minimum

temperatures may be partly explained by the influence of solar radiation, as previously mentioned and / or microclimatic differences between these locations. Regional data is not considered to be representative of local data but is indicative of the expected temperature trend.

A comparison of fruit fly trapping (details reported in Section 6.3) data and Perth Airport weather data for the trapping period in Figure 19 shows that fruit flies tend to be trapped soon after the onset of increasing daily maximum temperatures. This is consistent with observations reported in the literature review (Duyck & Quilici 2002), although based on a very small dataset.

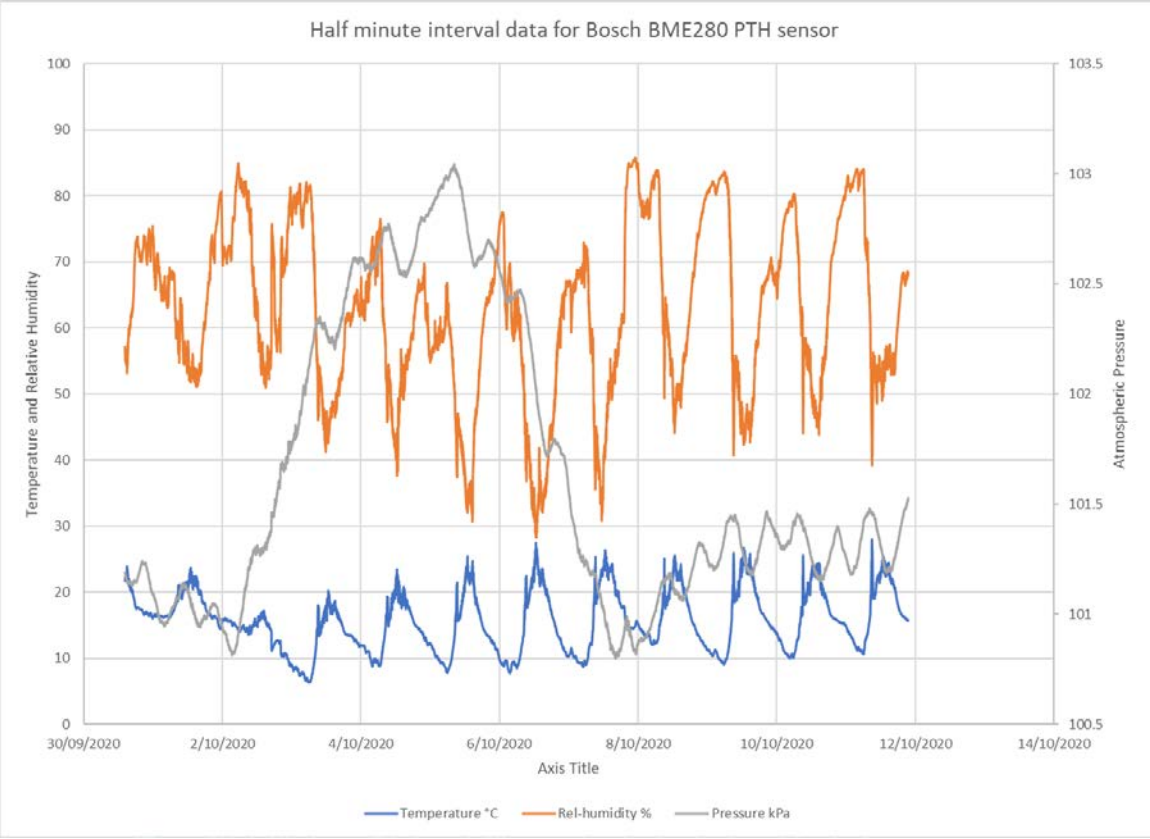


Figure 17 Graph of approximately 30 second interval PTH data from te Bosch BME280 sensor between 30 September 2020 and 11 October 2020.

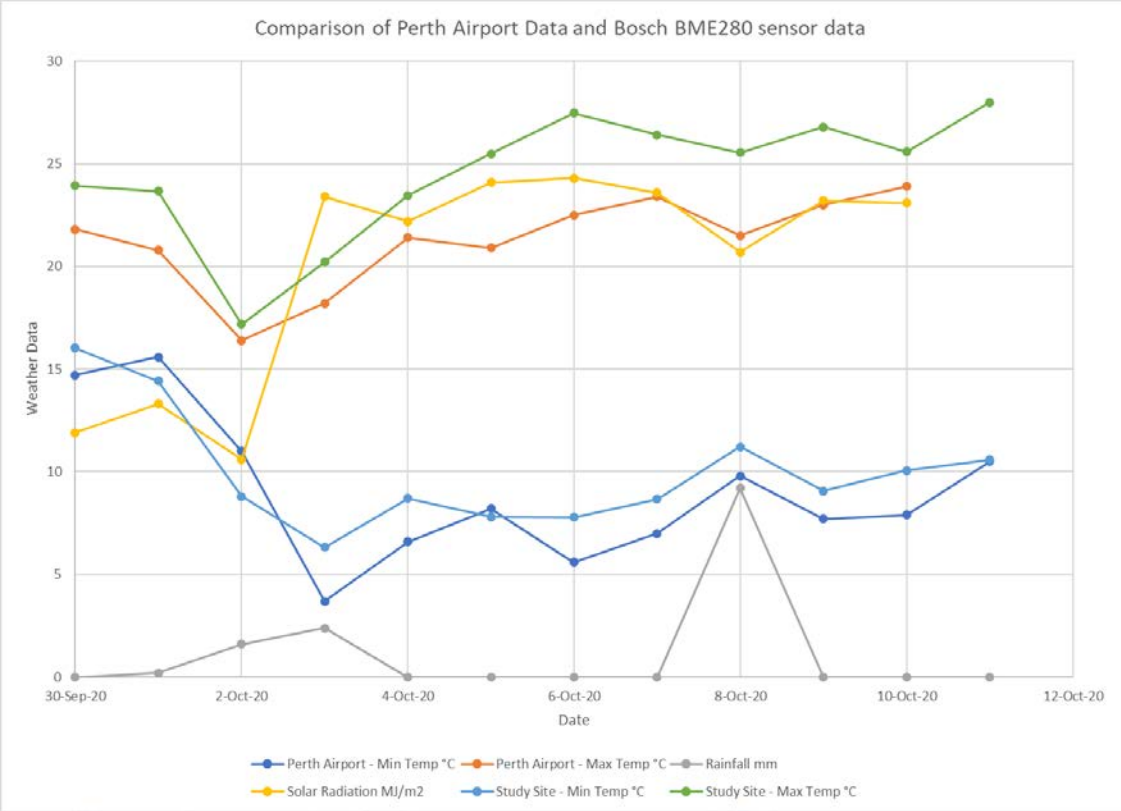


Figure 18 Comparison of daily weather data for Perth Airport and Bosch BME280 sensor data.

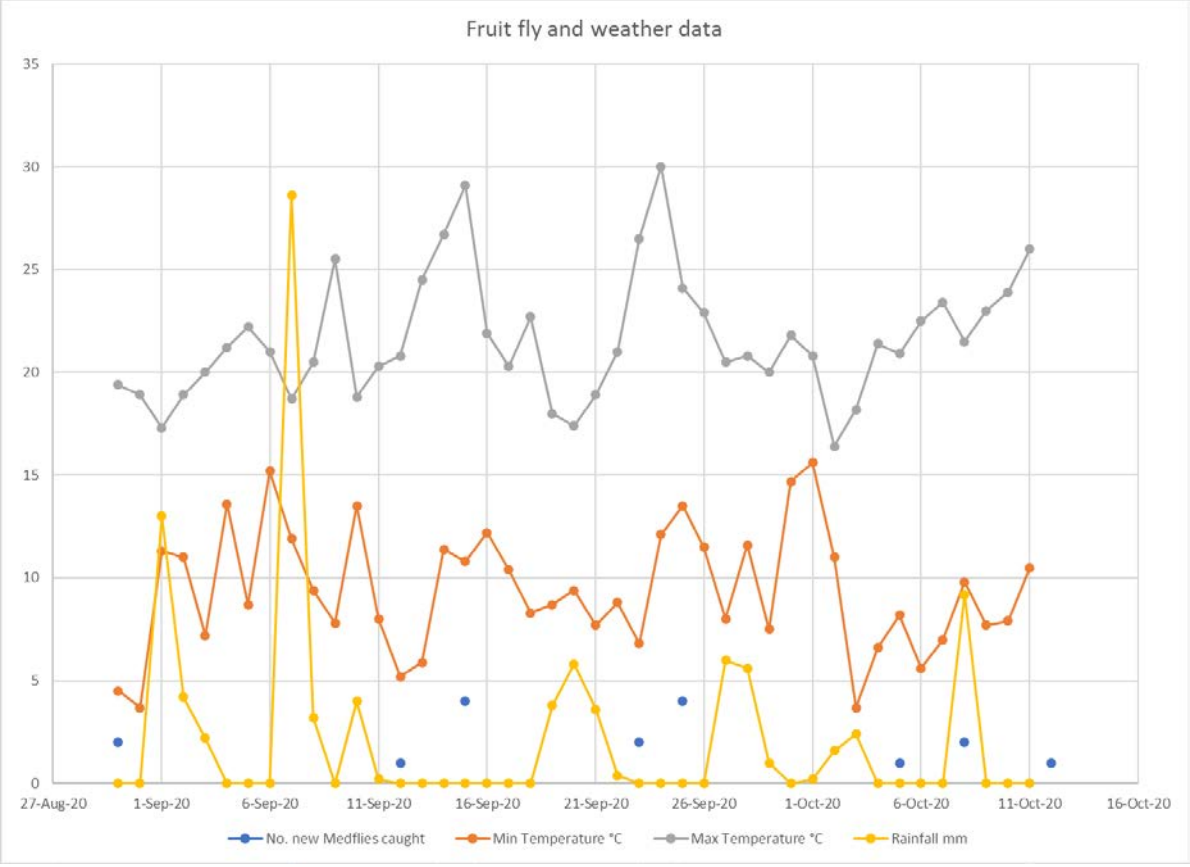


Figure 19 Perth Airport data (Bureau of Meteorology 2020) and Medfly trapping data comparison.

6.3 Medfly trapping

The trapping record for the test Medfly trap is included in Table 6. Trapping started late in the project due to a long delay in obtaining permission from USQ due to health and safety concerns associated with DDVP (Dichlorvos) insecticide cubes that are used with the Lynfield trap and are considered to be hazardous. The trap was hung on the 3 August 2020 but the first Medflies were only caught on 30 August 2020. This was consistent with expectations that few Medflies would be caught during the winter. During September and October more insects were trapped, as can be seen in Table 6 and Figure 20, with a total of 18 recorded by 12 October 2020. During rainy weather trap inspections were not done but the trap was checked on most dry days. Due to the lack of a functioning camera, the system could not be fully implemented to collect images on a remotely as intended.

Table 6 Record of fly trapping observations.

Date and time	Number of new Medflies	Total Medflies caught
30 August 2020 11:30	2	2
12 September 2020 17:40	1	3
13 September 2020 17:08	1	4
15 September 2020 16:40	4	8
23 September 2020 13:30	2	10
24 September 2020 15:10	0	10
25 September 2020 12:12	4	14
27 September 2020 13:41	0	14
30 September 2020 17:43	0	14
02 October 2020 13:11	0	14
05 October 2020 09:47	1	15
08 October 2020 09:28	2	17
10 October 2020 18:32	0	17
11 October 2020 18:34	0	17
12 October 2020 17:40	1	18

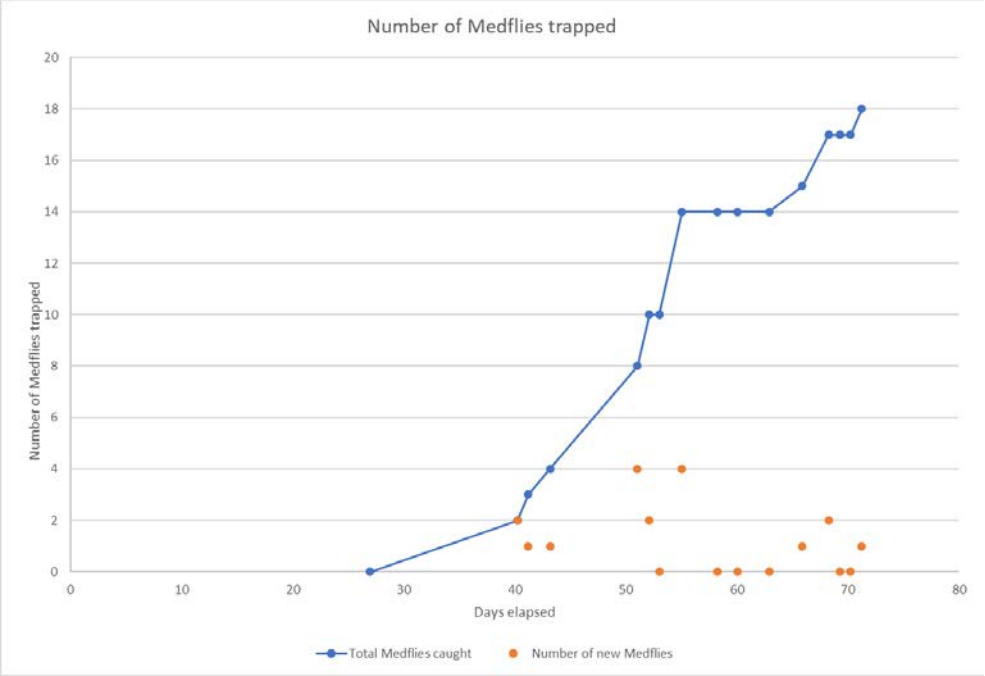


Figure 20 Medfly trapping data (monitoring event totals and cumulative totals).

6.4 Insect detection & identification

As discussed in section 5.5, the OpenCV Simple Blob Detector was used to detect Medflies after a process of elimination. Initial results using test flies provided for the project yielded relatively good success, as shown in Figure 21. No false negatives occurred and one false positive occurred due to the shadow of one Medfly after some trial and error changes in the settings for the Simple Blob Detector algorithm. The false positive was on a shadow of approximately the same size and hue as a fruit fly, as well as mimicking the shape of the fruit fly relatively accurately due to the angle of lighting. Further refinement of the imaging system and the trap may allow 100% discrimination if lighting conditions are suitable and repeatable, although species identification with a deep learning model is not possible until better images can be obtained and extracted for comparison.

During the project it was observed that false positives and negatives were more problematic than initial indications suggested, once other debris was included in the image and if illumination wasn't optimal. The last image taken of the field test trap was processed with the Simple Blob Detector again to illustrate likely performance under field conditions (with a clear trap background) and the results were less encouraging. There were 6 false negatives and 7 false positives, with a total of 12 Medflies correctly detected. The output of the Simple Blob Detector for this test is shown in Figure 22 and the settings used are captured in the modified blob detector algorithm in Appendix B.

Early in the project it was thought that better light control would be achieved with active lighting and the neopixel LED ring was implemented, however, back reflection was found to be a major

impediment to obtaining good quality images. In an attempt to eliminate this reflected light interference, a black background was considered to be a potential solution. It was thought that a black background would also mask the buildup of dark particulates that were observed in the test trap (visible in Figure 22) and which were producing false positives during testing of the Simple Blob Detector algorithm. Tests were done with black spray paint and black cardboard but these materials still resulted in excessive back reflection of light from the neopixel LEDs, as can be seen in Figure 23 and Figure 24.

Although it may still be possible to obtain a less reflective background with a better black coating, that will eliminate enough back reflection, it was decided to investigate the effectiveness of a deflection screen in front of the neopixel LED ring. A screen was designed as part of the equipment housing in Autodesk Fusion360, in the form of a cone (see Figure 25). The cone angle was designed at 70 degrees to ensure that it would not interfere with the field of view of the lens, particularly with respect to the target area of the trap. This angle was also expected to be effective at reflecting light off the sides onto the sides of the trap to produce a more diffuse and oblique indirect illumination source, if the screen's upper surface were sufficiently reflective (metallic coating or 3D printed in white).

Figure 21 Blob detection results for the same image as Figure 11 ***(Note blobs marked as fine red circles)****.*

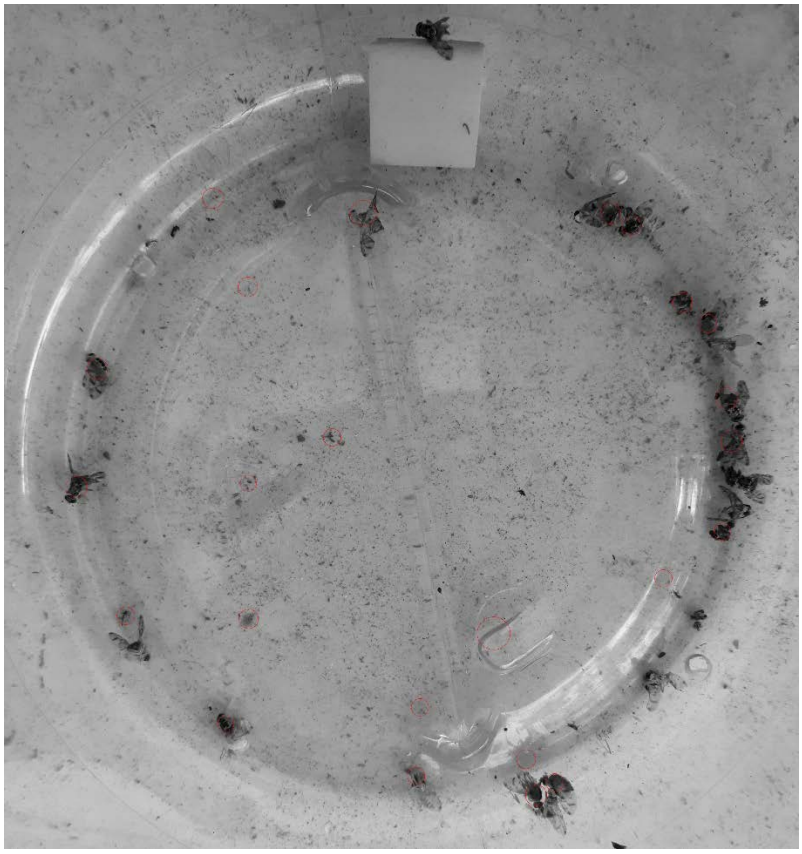


Figure 22 Final image of field test trap, processed with Simple Blob Detector. ***(Note blobs marked as fine red circles)****.*

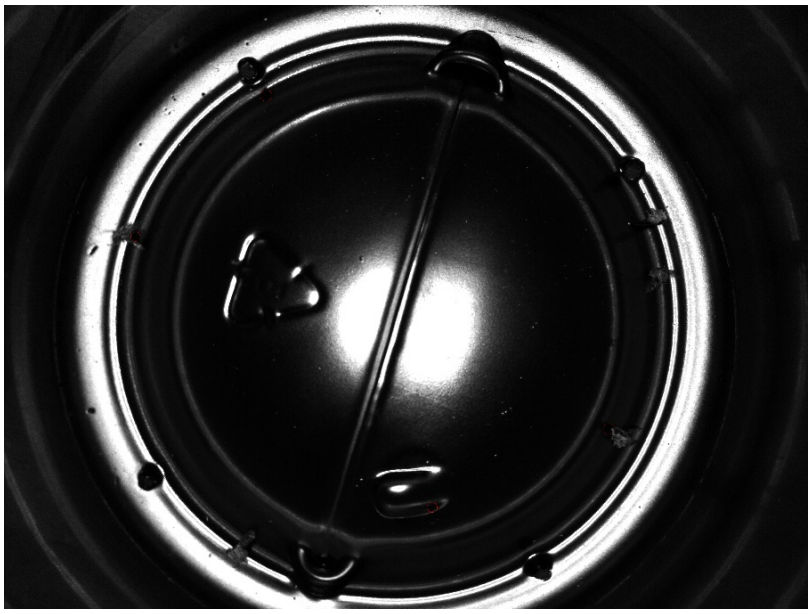


Figure 23 Black spray-painted base of Lynfield trap, still showing high back reflectance.

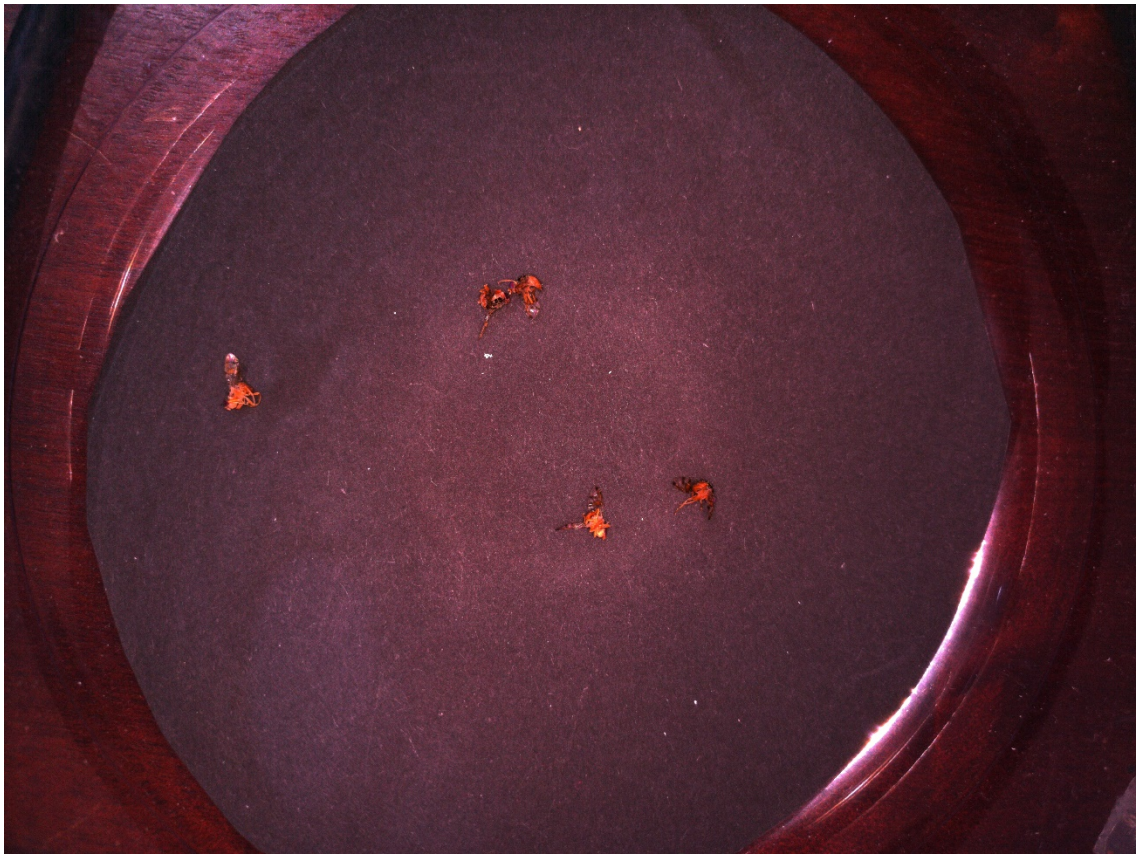


Figure 24 Lynfield trap lined with black cardboard, showing less reflection but similar luminosity to fruit flies in some areas.

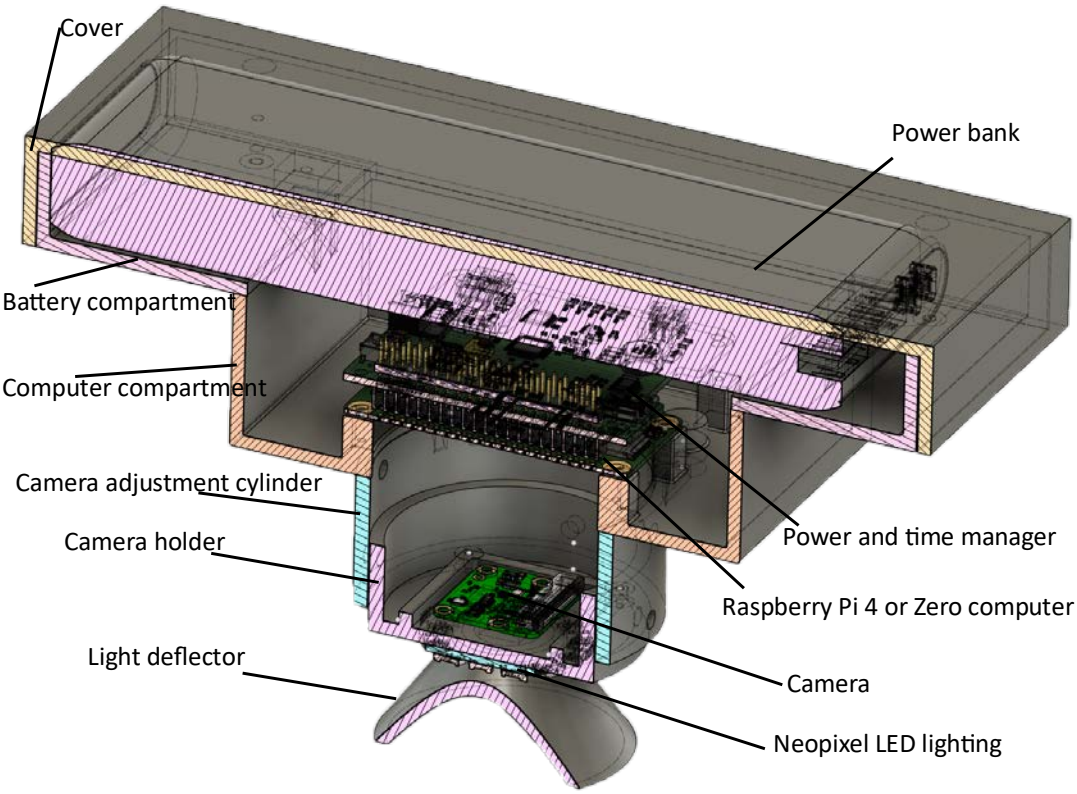


Figure 25 Revised design showing the light deflector below the neopixel ring light source.

Due to the poor selectivity of the Simple Blob Detector, other image processing alternatives were sought to identify fruit flies but with little success. An example of detection ability using the SIFT algorithm is show in Figure 26, where the green lines show matching of keypoints between the reference image (which was extracted from the detection image) and the detection image. Only the reference fruit fly was detected and others were not. Other image detection algorithms, such as SURF or BRISK, were also considered but the unsatisfactory performance of the SIFT algorithm for this work suggested that this would be futile, since the SIFT algorithm was found to be the overall best performer by Tareen & Saleem (2018). Notably the patent on the SIFT algorithm had recently expired, making it a viable option for open source software development if fit for purpose.

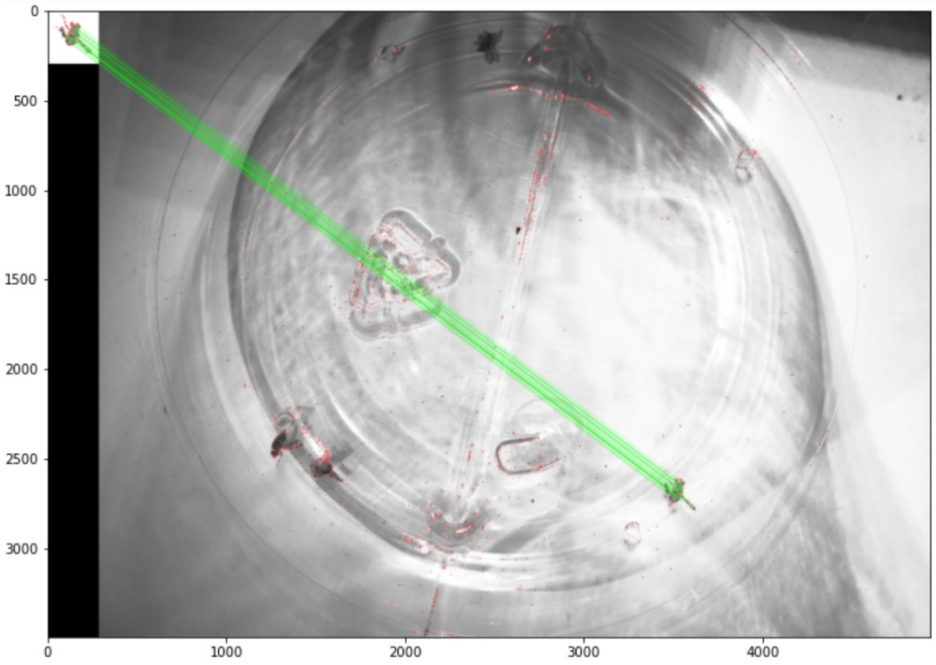


Figure 26 Evaluation of SIFT algorithm for Medfly detection, only identifying the fly used for the reference image.

The objective of automated species identification could not be achieved due to the reasons outlined in section 5.5. This was partly due to excessive time and effort expended on attempting to obtain useful images from the Arducam IMX298 camera but also due to the reasons previously stated with respect to implementation of the algorithm developed by Valan et al. (2019). The inherent challenges of learning Python, OpenCV and Tensorflow simultaneously to the rest of the project and having to update software written for substantially different versions of these software platforms also caused delays.

For the training dataset intended to train the deep learning model of Valan et al. (2019), a total of 1421 fruit fly images were extracted from a larger dataset, of which 711 were of Medflies. Due to the labour-intensive process of preparing this dataset, its preparation was stopped when it was realised that the dataset would still be inadequate and that removal of duplicates

would reduce its size further. The lack of diversity in the angles from which flies were photographed or poses that would be suitable for identifying dead flies, also meant that the training set would probably be a poor training dataset and that effort would be better spent elsewhere for the remainder of the project.

7 Conclusions

The limitations of the Arducam IMX 298 camera and difficulties with integration of Python 2 and Python 3 code, as well as incompatibility of some code between different versions of OpenCV and Tensorflow created various complications. Difficulty in obtaining parts due to COVID-19, both due to product scarcity and freight delays, was also problematic.

Despite these limitations, the PTH sensor was successfully implemented and a communication algorithm was also created to email collated PTH and Medfly count results to a designated email address, which could be used in practice as a data collection strategy. This data format requires low-bandwidth and little additional programming and could use either WiFi, where feasible, or LoRa or Narrow Band IoT protocols for more geographically dispersed systems, which would be typical in large-scale horticulture and agriculture.

The power and time management hat for the Raspberry Pi could not be implemented in the available time. The lack of an English manual contributed to this. The Raspberry Pi's time was set over the WiFi system nonetheless, allowing the system to function as intended to collect continuous weather data. It was not possible to shut down and restart the Raspberry Pi computer without the power and time management hat. With additional time it is expected that the power and time management functions will be successfully implemented as intended without excessive complications.

Despite the limitations, a partially successful implementation of the aims and objectives of the project were achieved. Reasonable discrimination of Medflies from background was achieved from sample photos using the OpenCV Simple Blob Detector algorithm. Although accuracy is limited, this method is considered to be potentially useful to search for fruit flies if it is allowed to generate more false positives and no false negatives. In such an instance blobs could then be extracted as individual image frames to then be fed through a convolutional neural network based algorithm for positive identification. This approach would have to be verified and optimised experimentally.

In hindsight the project aims and objectives were ambitious, given the learning requirements involved and the time available, particularly considering all the technical and logistical complications which occurred, including significant delays in obtaining parts due to COVID-19

freight delays and shortages of items, as well as slow responses from Arducam regarding camera setup. Despite the setbacks, further effort to develop the project in the future would still be considered worthwhile given its potential benefits.

8 Recommendations

Prior to expending further effort on development and implementation, a fully functional identification algorithm should be tested against images taken with a more reliable 16MP camera. This should be done with images taken against a black background with optimal illumination, to ensure that a 16MP camera captures adequate detail for definitive identification. This is the major potential flaw in the conceptual system design and needs to be eliminated before additional time and expense can be justified. There was insufficient time during the project to pursue this option, although images taken with the Samsung A7 2017 mobile phone would be a suitable benchmark using a suitably configured trap and lighting.

Future development of the project is strongly dependent on obtaining a suitable camera and control software to collect suitable quality images remotely. Through communication with Arducam it was revealed that a new USB video device class (UVC) version of the IMX298 camera would be released and that it would be capable of the image processing functions that the current model (used for the project) lacks. Consideration should be given to evaluating this model when it is released, as well as searching for other alternatives as new models are released by different manufacturers, provided that 16MP images are demonstrated to be adequate for identification.

If the above limitations can be overcome, LoRa technology should be evaluated as a potential communication protocol, as it generally conforms the design philosophy of the project. Since LoRa is a non-subscription radio communication system and transmission equipment is available at competitive costs, as well as software being open source, it is considered to be a strong candidate for meeting all project objectives if implemented with the rest of the platform. Implementation of LoRa would likely require modifications to the housing design, depending on which hat is selected, although they would likely not be major.

A suitable power and time management hat should be implemented to enable automated battery charging and real time clock operations, possibly a cheaper model than the S.USV model obtained for development. Addition of a suitable solar charging capability would be necessary for field implementation and could be integrated through this system. Sizing of a solar panel would depend on the overall energy requirements for standby, data collection, processing and transmission. These specifications could only be determined reliably once the

rest of the hardware and software is fully functional, as processing time will be a major factor in determining daily energy requirements.

9 References

ABC - Gardening Australia 2015, *Fighting Fruit Fly*, viewed 28 March 2020, <https://www.abc.net.au/gardening/factsheets/fighting-fruit-fly/9436984>.

ABC - Rural 2015, *Fruit fly trap finds effective lure for female flies*, viewed 28 March 2020, <https://www.youtube.com/watch?v=g2Ki7ZTgneQ>.

Adafruit forum 2016, *BME 280 temp offset/self heating*, viewed 14 August 2020, <https://forums.adafruit.com/viewtopic.php?p=534525>.

Al-Behadili, FJ, Bilgi, V, Li, J, Wang, P, Taniguchi, M, Agarwal, M, Ren, Y & Xu, W 2019, 'Cold Response of the Mediterranean Fruit Fly (Ceratitis capitata) on a Lab Diet', *Insects*, vol 48, no. 10, pp. 1-11.

Allwood, AJ & Leblanc, L 1997, 'Losses Caused by Fruit Flies (Diptera: Tephritidae) in Seven Pacific Island Countries', Nadji, Fiji.

Arbuckle, T, Schröder, S, Steinhage, V & Wittmann, D 2001, 'Biodiversity Informatics in Action: Identification and Monitoring of Bee Species using ABIS', *EnviroInfo 2001*, Metropolis Verlag, Marburg.

Arducam 2019, *Cameras for Raspberry Pi*, viewed 20 May 2020, <https://www.arducam.com/docs/cameras-for-raspberry-pi/>.

Arducam 2020, *Camera Userland Driver SDK and Examples*, viewed 19 June 2020, <https://www.arducam.com/docs/cameras-for-raspberry-pi/mipi-camera-modules/camera-userland-driver-sdk-and-examples/>.

Australian Interstate Quarantine 2019, *Major Pests*, viewed 13 April 2020, <https://www.interstatequarantine.org.au/major-pests/>.

Australian Interstate Quarantine 2019, *Mediterranean Fruit Fly*, viewed 22 May 2020, <https://www.interstatequarantine.org.au/pest/mediterranean-fruit-fly/>.

Beavis, L 2018, *Tasmania's fruit fly fight hits $5.5m bill, but Minister espects victory in months ahead*, viewed 26 May 2020, <https://www.abc.net.au/news/2018-05-24/fruit-fly-infestation-tasmania-cost-revealed/9791898>.

Bebber, DP, Ramotowski, MA & Gurr, SJ 2013, 'Crop pests and pathogens move polewards in a warming world', *Nature Climate Change Letters*, pp. 1-4.

Biotrap 2020, *Biotrap V2 X*, viewed 14 April 2020, <https://www.biotrap.com.au/directory/products/23/21/>.

BioTrap 2020, *Delta Trap*, viewed 22 May 2020, <https://www.biotrap.com.au/directory/products/23/18/>.

Biotrap 2020, *Jackson Trap*, viewed 8 May 2020, <https://www.biotrap.com.au/directory/products/23/20/>.

BioTrap 2020, *Lynfield Trap*, viewed 21 May 2020, <https://www.biotrap.com.au/directory/products/23/15/lynfield-trap/>.

Broughton, S 2018, *Fruit Fly Monitoring in Commercial Orchards*, viewed 28 March 2020, <https://www.agric.wa.gov.au/medfly/fruit-fly-monitoring-commercial-orchards>.

Broughton, S 2018, *Mediterranean fruit fly life cycle & biology*, viewed 28 March 2020, <https://www.agric.wa.gov.au/medfly/mediterranean-fruit-fly-life-cycle-biology>.

Broughton, S 2019, *Piloting new techniques to control and eradicate Mediterranean fruit fly in Carnarvon*, viewed 8 May 2020, <https://www.agric.wa.gov.au/medfly/piloting-new-techniques-control-and-eradicate-mediterranean-fruit-fly-carnarvon>.

Broughton, S 2020, *Queensland Fruit Fly*, viewed 28 March 2020, <https://www.agric.wa.gov.au/plant-biosecurity/queensland-fruit-fly>.

Bugs for Bugs, *Fruit fly trap*, viewed 13 April 2020, <https://bugsforbugs.com.au/product/fruit-fly-trap/>.

Bureau of Meteorology 2020, *Climate Data Online*, viewed 12 October 2020, <http://www.bom.gov.au/climate/data/>.

Business Queensland 2019, *Mediterranean fruit fly*, viewed 13 April 2020, <https://www.business.qld.gov.au/industries/farms-fishing-forestry/agriculture/crop-growing/priority-pest-disease/mediterranean-fruit-fly>.

Business Queensland 2019, *Oriental Fruit Fly*, viewed 28 March 2020, <https://www.business.qld.gov.au/industries/farms-fishing-forestry/agriculture/crop-growing/priority-pest-disease/oriental-fruit-fly>.

Calver, O 2019, *Trapview allows pests to be monitored through your phone*, viewed 8 May 2020, <https://www.theland.com.au/story/6315471/pest-monitor-predicts-insect-flight-times/>.

Cheng, X, Zhang, Y, Chen, Y, Wu, Y & Yue, Y 2017, 'Pest identification via deep residual learning in complex background', *Computers and Electronics in Agriculture*, vol 141, pp. 351-356.

Cherry Growers Australia Inc 2014, *Pest Fact Sheet - Fruit Fly*, viewed 28 March 2020, <https://www.utas.edu.au/__data/assets/pdf_file/0016/1110391/IPDM-BINDER-pest-sheets.pdf>.

Chulu, F, Phiri, J, Nkunika, PO & Yirenda, M 2019, 'A Convolutional Neural Network for Automatic Identification and Classification of Fall Army Worm Moth', *IJACSA*, vol 10, no. 7, pp. 112-118.

Cook, DC & Fraser, RW 2014, 'Eradication versus control of Mediterranean fruit fly in Western Australia', *Agricultural and Foroest Entomology*, pp. 1-8.

Core Electronics 2019, *How To Setup Raspberry Pi Zero W Headless WiFi*, viewed 25 May 2020, <https://core-electronics.com.au/tutorials/raspberry-pi-zerow-headless-wifi-setup.html>.

Cros, M-J, Aubertot, J-N, Gaba, S, Xavier , , Sabbadin, R & Nathalie, P 2002, 'Improving Pest Monitoring Networks in order to reduce pesticide use in agriculture'.

CSIRO 2020, *Integrated Pest Management (IPM)*, viewed 12 August 2020, <https://www.csiro.au/en/Research/AF/Areas/Sustainable-farming-systems/Integrated-Pest-Management>.

Das, A 2020, *Install pyTorch in Raspberry Pi 4 (or any other)* , viewed 20 May 2020, <https://gist.github.com/akaanirban/621e63237e63bb169126b537d7a1d979>.

Daylilies in Australia 2016, *Fruit Fly Control – Make Your Own Fruit Fly Traps*, viewed 28 March 2020, <https://www.dayliliesinaustralia.com.au/fruit-fly-control-for-gardeners/>.

De Lima, CP, Jessup, AJ, Mansfield, ER & Daniels, D 2011, 'Cold treatment of table grapes infested with Mediterranean fruit fly Ceratitis capitata (Wiedemann) and Queensland fruit fly Bactrocera tryoni (Froggatt) Diptera: Tephritidae', *New Zealand Journal of Crop and Horticultural Science*, vol 39, no. 2, pp. 95-105.

De Meyer, M, Robertson, MP, Peterson, AT & Mansell, MW 2008, 'Ecological niches and potential geographical distributions of Mediterranean fruit fly (Ceratitis capitata) and Natal fruit fly (Ceratitis rosa)', *J. Biogeogr.*, vol 35, pp. 270-281.

Department of Agriculture and Fisheries - Queensland Government 2017, *A-Z list of horticultural insect pests*, viewed 13 April 2020, <https://www.daf.qld.gov.au/business-priorities/agriculture/plants/fruit-vegetable/insect-pests>.

Department of Agriculture, Water and the Environment - Australian Government 2019, *Top 40 Exotic and Unwanted Plants and Pests*, viewed 13 April 2020, <https://www.agriculture.gov.au/pests-diseases-weeds/plant>.

Department of Primary Industries and Regional Development - Western Australia 2020, *Fall armyworm: declared pest*, viewed 14 April 2020, <https://www.agric.wa.gov.au/plant-biosecurity/fall-armyworm-declared-pest>.

Department of Primary Industries and Regional Development - Western Australia, *Pests*, viewed 13 April 2020, <https://www.agric.wa.gov.au/pests-weeds-diseases/pests>.

Department of Primary Industries and Regional Development 2020, *Biosecurity alerts: Queensland fruit fly updates*, viewed 13 April 2020, <https://www.agric.wa.gov.au/plant-biosecurity/biosecurity-alerts-queensland-fruit-fly-updates>.

Dionysopoulou, NK, Papanastasiou, SA, Kyritsis, GA & Papadopoulos, NT 2020, 'Effect of host fruit, temperature and Wolbachia infection on survival and development of Ceratitis capitata immature stages', *Plos One*, vol 15, no. 3, pp. 1-19.

DTN 2020, *DTN Agronomic Platform*, viewed 11 May 2020, <https://www.dtn.com/agriculture/agribusiness/dtn-agronomic-platform/#>.

Duckett, C 2018, *CSIRO Innovation Fund backs technology to tackle fruit flies*, viewed 9 May 2020, <https://www.zdnet.com/article/csiro-innovation-fund-backs-technology-to-tackle-fruit-flies/>.

Duncan, P 2019, *TensorFlow on a Raspberry Pi Zero is a Bad Idea*, viewed 11 May 2020, <https://pablotron.org/2019/04/06/tensorflow-on-a-raspberry-pi-zero-w-is-a-bad-idea.html>.

Duyck, PF & Quilici, S 2002, 'Survival and development of different life stages of three Ceratitis spp. (Diptera: Tephritidae) reared at five constant temperatures', *Bulletin of Entomological Research*, vol 92, pp. 461-469.

Fruit Fly Identification Australia 2020, *Fruit Fly ID Australia*, viewed 28 March 2020, <https://fruitflyidentification.org.au/identify/>.

Gepro Distributors 2016, *Mediterranean Fruit Fly Trap*, viewed 14 April 2020, <https://www.gepro.com.au/products/mediterranean-fruit-fly-trap/>.

GitHub 2020, *ArduCAM/MIPI_camera*, viewed 19 June 2020, <https://github.com/ArduCAM/MIPI_Camera/tree/master/RPI>.

Glick, J & Miller, K 2016, 'Insect ClassificationWith Heirarchical Deep Convolutional Neural Networks Convolutional Neural Networks for Visual Recognition (CS231N)', Stanford University, Stanford.

Goodfellow, I, Bengio, Y & Courville, A 2016, *Deep Learning*, MIT Press, <www.deeplearningbook.org>.

Gutierrez, P 2017, *Connected fly traps giving insights to growers*, viewed 13 April 2020, <https://www.iothub.com.au/news/connected-fly-traps-giving-insights-to-growers-455853>.

Hafi, A, Arthur, T, Symes, M & Millist, N 2013, 'Benefit-cost analysis of the long term containment strategy for exotic fruit flies in the Torres Strait', Australian Government, Department of Agriculture, ABARES, Canberra.

Hansen, OL, Svenning, J-C, Olsen, K, Dupont, S, Garner, BH, Iosifidis, A, Price, BW & Høye, TT 2020, 'Species-level image classification with convolutional neural network enables insect identification from habitus images', *Ecology and Evolution*, vol 10, pp. 737-747.

Hendrichs, J & Robinson, A 2009, 'Sterile Insect Technique', in VH Resh, RT Cardé (eds.), *Encyclopedia of Insects (Second Edition)*, Elsevier Inc, New York.

Hernández-Serna, A & Jiménez-Segura, LF 2014, 'Automatic identification of species with neural networks', *PeerJ*, pp. 1-16.

International Plant Protection Convention 2014, *Four devastating fruit flies pests are one and the same species*, viewed 12 August 2020, <https://www.ippc.int/en/news/four-devastating-fruit-flies-pests-are-one-and-the-same-species/>.

Johnson, K 2019, *Facebook launches PyTorch Mobile for edge ML on Android and iOS devices*, viewed 20 May 2020, <https://venturebeat.com/2019/10/10/facebook-launches-pytorch-mobile-for-edge-ml-on-android-and-ios-devices/>.

Kanda, N 2018, *How to install PyTorch on the Raspberry Pi 3B+ (RPI)*, viewed 20 May 2020, <https://github.com/NikhilKanda/Pytorch-on-Raspberry-PI-3B/blob/master/How%20to%20install%20PyTorch%20on%20Raspberry%20Pi%203B.md>.

Khan, A, Sohail, A, Zahoora, U & Qureshi, AS 2020, 'A survey of the recent architectures of deep convolutional neural networks', *Artificial Intelligence Review*, vol Online.

Krizhevsky, A, Nair, V & Hinton, G 2015, *Alex Krizhevsky's home page*, viewed 11 May 2020, <https://www.cs.toronto.edu/~kriz/cifar.html>.

Kurama, V 2020, *PyTorch vs. TensorFlow: Which Framework Is Best for Your Deep Learning Project?*, viewed 22 May 2020, <https://builtin.com/data-science/pytorch-vs-tensorflow>.

Lupori, A 2020, *Ubidots*, viewed 20 May 2020, <https://ubidots.com/blog/nb-iot-vs-lte-m/>.

MathWorks 2020, *What is Deep Learning?*, viewed 12 May 2020, <https://au.mathworks.com/discovery/deep-learning.html>.

Matt 2016, *Using the BME280 I2C Temperature and Pressure Sensor in Python*, viewed 14 August 2020, <https://www.raspberrypi-spy.co.uk/2016/07/using-bme280-i2c-temperature-pressure-sensor-in-python/>.

McIntyre, K 2016, *Smart traps to provide pest early warning*, viewed 20 April 2020, <https://grdc.com.au/resources-and-publications/groundcover/ground-cover-issue-121-mar-apr-2016/smart-traps-to-provide-pest-early-warning>.

Meats, A & Smallridge, CJ 2007, 'Short- and long-range dispersal of medfly, Ceratitis capitata (Dipt., Tephritidae), and its invasive potential', *J. Appl. Entomol.*, vol 131, no. 8, pp. 518–523.

Miranda, JL, Gerardo, BD & Tanguilig, BTI 2014, 'Pest Identification using Image Processing Techniques in Detecting Image Pattern through Neural Network', *IJIPT*, vol 1, no. 4, pp. 4-9.

Motta, D, Santos, AÁB, Winkler, I, Aparecida, B, Machado, S, Pereira, DADI, Cavalcanti, AM, Fonseca, EOL, Kirchner, F & Badar 2019, 'Application of convolutional neural networks for classification of adult mosquitoes in the field', *Plos One*, vol 14, no. 1, pp. 1-18.

Motta, D, Santos, AÁB, Winkler, I, Aparecida, B, Machado, S, Pereira, DADI, Cavalcanti, AM, Fonseca, EOL, Kirchner, F & Badaró, R 2019, 'Application of convolutional neural networks for classification of adult mosquitoes in the field', *Plos One*, vol 14, no. 1, pp. 1-18.

Nicolai, C 2016, *Python-BME280*, viewed 25 September 2020, <https://github.com/cmur2/python-bme280>.

Nicolai, C 2020, *GitHub cmur2 / python-bme280*, viewed 14 August 2020, <https://github.com/cmur2/python-bme280>.

Nine Network 2012, *The Garden Gurus*, viewed 28 March 2020, <https://www.youtube.com/watch?v=g2Ki7ZTgneQ>.

Nvidia, *Train Models Faster*, viewed 12 May 2020, <https://developer.nvidia.com/cuda-zone>.

OpenCV 2020, *About*, viewed 12 May 2020, <https://opencv.org/about/>.

Organic Crop Protectants 2019, *eco-lure male Queensland fruit fly trap*, viewed 14 April 2020, <https://ecoorganicgarden.com.au/products/pest-disease/eco-lure-male-queensland-fruit-fly-trap/>.

Palmer, K 2017, *Fruit fly trial in Western Australia*, viewed 27 May 2020, <https://www.agric.wa.gov.au/fruit-fly-trial-western-australia>.

Paperswithcode 2020, *Image Classification on CIFAR-100*, viewed 12 May 2020, <https://paperswithcode.com/sota/image-classification-on-cifar-100>.

Paranhos, BJ, Papadopoulos, N, McInnis, , Gava, C, Lopes, FSC, Renata, M & Malavasi, A 2010, 'Field Dispersal and Survival of Sterile Medfly Males Aromatically Treated with Ginger Root Oil', *Environmental Entomology*, vol 39, no. 2, pp. 570-575.

PIRSA 2020, *Protecting SA from fruit fly*, viewed 8 May 2020, <https://pir.sa.gov.au/biosecurity/fruit_fly_in_sa/how_you_can_stop_fruit_fly/what_pirsa_does/Protection_and_prevention>.

Plant Health Australia 2020, *National Fruit Fly Council*, viewed 26 May 2020, <https://portal.biosecurityportal.org.au/Pages/NFFCLanding.aspx>.

Plant Health Australia, *Fruit flies*, viewed 27 May 2020, <https://www.planthealthaustralia.com.au/national-programs/fruit-fly/>.

Plant Health Australia n.d.b, *National Fruit Fly Strategy*, viewed 26 May 2020, <https://www.planthealthaustralia.com.au/national-programs/fruit-fly/draft-national-fruit-fly-strategy/>.

Precision Farming Dealer 2016, *Spensa Tech Z-Trap Insect Trap*, viewed 11 May 2020, <https://www.precisionfarmingdealer.com/articles/2531-spensa-tech-z-trap-insect-trap>.

Prevent fruit fly 2017, *Smart traps – a new tool in fruit fly management*, viewed 13 April 2020, <https://preventfruitfly.com.au/smart-traps-a-new-tool-in-fruit-fly-management/>.

Prevent Fruit Fly 2019, *Trapping*, viewed 28 March 2020, <https://preventfruitfly.com.au/trapping/>.

Prevent Fruit Fly 2020, *Prevent Fruit Fly*, viewed 26 May 2020, <https://preventfruitfly.com.au/>.

Q-engineering 2020, *Deep learning with Raspberry Pi and alternatives in 2020*, viewed 8 May 2020, <https://qengineering.eu/deep-learning-with-raspberry-pi-and-alternatives.html>.

Rahman, T & Broughton, S 2019, 'The Survival of Mediterranean Fruit Fly (Diptera: Tephritidae) Over Winter in Western Australia', *Environmental Entomology*, vol 48, no. 4, pp. 977-987.

RapidAIM 2018, *RapidAIM*, viewed 13 April 2020, <https://rapidaim.io/>.

Raspberry Pi 2020, *Setting up your Raspberry Pi*, viewed 6 June 2020, <https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up>.

Rembor, K 2020, *Raspberry Pi Wiring*, viewed 5 July 2020, <https://learn.adafruit.com/neopixels-on-raspberry-pi/raspberry-pi-wiring>.

Russell, M 2020, *The game-changing technology helping with pest detections*, viewed 22 May 2020, <https://www.freshplaza.com/article/9197998/the-game-changing-technology-helping-with-pest-detections/>.

Schellhorn, N 2020, *Agtech drives pesticide use efficiencies*, viewed 9 May 2020, <https://apal.org.au/rapidaim-agtech-pesticide-efficiency/>.

Sciarretta, A, Tabilio, MR, Lampazzi, E, Ceccaroli, C, Colacci, M & Trematerra, P 2018, 'Analysis of the Mediterranean fruit fly [Ceratitis capitata (Wiedemann)] spatio-temporal distribution in relation to sex and female mating status for precision IPM', *Plos One*, vol 13, no. 4, pp. 1-23.

Searles Garden Products 2017, *Searles fruit fly trap*, viewed 14 April 2020, <https://www.searlesgardening.com.au/products/category/RNWDXERX-fruit-fly/SFFTRAP--searles-fruit-fly-trap>.

Semtech, *What is LoRa*, viewed 12 May 2020, <https://www.semtech.com/lora/what-is-lora>.

Simonyan, K & Zisserman, A 2015, 'Very Deep Convolutional Networks for Large-Scale Image Recognition', *ICLR 2015 proceedings*, San Diego.

Smith, R 2018, *Wide range of Hygrometers:*, viewed 8 October 2020, <http://www.kandrsmith.org/RJS/Misc/Hygrometers/calib_many.html>.

Snaptrap 2017, *Remote Trap Monitoring*, viewed 16 April 2020, <https://snaptrap.com.au/>.

Stöffelbauer, A 2020, *Machine Learning in TensorFlow vs PyTorch*, viewed 20 April 2020, <https://towardsdatascience.com/pytorch-vs-tensorflow-in-code-ada936fd5406>.

Strachan, NJ & Nesvadba, P 1990, 'Fish Species Recognition by Shape Analysis of Images', *Pattern Recognition*, vol 23, no. 5, pp. 539-544.

Sun, Y, Liu, X, Yuan, M, Ren, L, Wang, J & Chen, Z 2018, 'Automatic in-trap pest detection using deep learning for pheromone-based Dendroctonus valens monitoring', *Biosystems Engineering*, vol 176, pp. 140-150.

Tareen, SAK & Saleem, Z 2018, 'A Comparative Analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK', 2018 International Conference on Computing, Mathematics and Engineering Technologies – iCoMET 2018, Sukkur, Pakistan.

Tensorflow 2020, *TensorFlow Lite for Microcontrollers*, viewed 12 May 2020, <https://www.tensorflow.org/lite/microcontrollers>.

The Goulburn Murray Valley (GMV) Regional Fruit Fly Group 2019, *East and West Collaboration to Fight Queensland Fruit Fly (10th Dec 2019)*, viewed 26 May 2020, <https://gmv-qldfruitfly.com.au/event/east-and-west-collaboration-to-fight-queensland-fruit-fly-10th-dec-2019/>.

The Things Network 2020, *Limitations of LoRa WAN*, viewed 20 May 2020, <https://www.thethingsnetwork.org/docs/lorawan/limitations.html>.

Thenmozhi, K & Srinivasulu Reddy, U 2019, 'Crop pest classification based on deep convolutional neural network and transfer learning', *Computers and Electronics in Agriculture*, vol 164, pp. 1-11.

Trapview 2013, *TRAPVIEW*, viewed 28 April 2020, <https://www.trapview.com/en/>.

Ubiquiti 2020, *All Products*, viewed 20 May 2020, <https://www.ui.com/products/#default>.

Valan, M, Makonyi, K, Maki, A, Vondráček, D & Ronquist, F 2019, 'Taxonomic Identification of Insects with Expert-Level Accuracy Using Effective Feature Transfer from Convolutional Networks', *Syst. Biol*, vol 68, no. 6, pp. 876-895.

van Driesche, R, Hoddle, M & Center, T 2008, *Control of Pests and Weeds by Natural Enemies*, Blackwell Publishing, Malden, MA.

von Hörchner, C 2018, *A backyard invention may spell curtains for fruit fly*, viewed 16 April 2020, <https://www.abc.net.au/news/rural/2018-06-22/fruit-fly-pest-invention-uses-mobile-phone-technology/9891420>.

Wang, J, Lin, C & Liang, A 2012, 'A new automatic identification system of insect images at the order level', *Knowledge-Based Systems*, vol 33, pp. 102-110.

Wen, C & Guyer, D 2012, 'Image-based orchard insect automated identification and classification method', *Computer and Electonics in Agriculture*, vol 89, pp. 110-115.

Xia, D, Chen, P, Wang, B, Zhang, J & Xie, C 2018, 'Insect Detection and Classification Based on an Improved Convolutional Neural Network', *Sensors*, vol 18, pp. 1-12.

Yang, H-P, Ma, C-S, Wen, H, Zhan, Q-B & Wang, X-L 2015, 'A tool for developing an automatic insect identification system based on wing outlines', *Nature Scientific Reports*, vol 5:12786, pp. 1-11.

Appendix A

Project Specification

AUTOMATED IDENTIFICATION OF INSECT PESTS IN TRAPS

ENG4111/4112 Research Project

Project Specification

For: Pierre Rousseau

Title: Automatic identification of insect pests in traps

Major: Mechatronic Engineering

Supervisors: Craig Lobsey

Enrollment: ENG4111 – EXT S1, 2020

ENG4112 – EXT S2, 2020

Project Aim: To develop a machine vision system for automatic detection of insects in pest control traps.

Programme: Version 1, 18th March 2020

1. Conduct initial background research on beneficial, exotic and endemic insect pests of interest to Australian agriculture, biosecurity, and ecosystems. Identify a suitable target species and application.
2. Review existing trapping systems, detection systems and machine vision hardware and software technology.
3. Conceptualise a suitable configuration for insect trapping and detection and establish technical requirements.
4. Assess hardware requirements for necessary capability and costs.
5. Select hardware and a suitable software development environment.
6. Construct an initial prototype to facilitate data collection.
7. Develop a machine vision algorithm for insect counting / identification.
8.
9. Deploy the prototype and algorithms at a suitable location and record data for evaluation
10. Process and evaluate experimental data.

If time and resource permit:

11. Refine detection and data processing algorithms, depending on what is achieved earlier.
12. Develop more sophisticated (possibly wireless) download options.

Appendix B

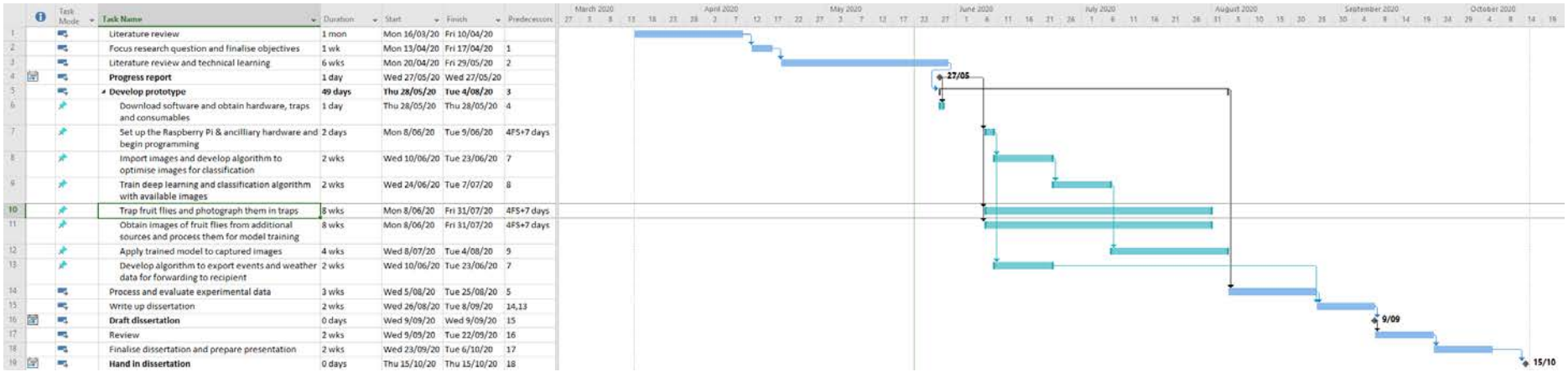
Workflow & Schedule and Risk Assessment

B.1
WORKFLOW AND SCHEDULE

The proposed workflow is summarised below:

- Obtain a Raspberry Pi Zero computer, camera and temperature / relative humidity sensor and assemble them.
- Download all required software and upload and initialise it on the Raspberry Pi.
- Download training images from the internet.
- Develop hardware integration and data capture algorithms.
- Implement algorithms to extract individual insect images and optimise them for image classification purposes using Python and OpenCV. It was envisaged that this would include some background removal, resizing to appropriate dimensions and enhancement of lighting. Other algorithms that could improve classification performance would also be considered.
- Import optimised images for model training.
- Process available images of Medfly, and potentially other images, to train the automated image classification algorithm of Valan et al. (2019). Make changes to the algorithm if necessary.
- Design the housing and include additional electronic components, such as LED illumination to be used within the lid of the Lynfield trap to ensure good quality photographs.
- Implement an algorithm to record insect counts and temperature, relative humidity and pressure data.
- Optional GPS and power management hats may be attached if time permits, with a view to developing the unit for remote deployment.
- If time permits, add data transmission hardware and implement an appropriate data transfer protocol on the selected hardware, to transmit collected data to a central computer.
- Add additional design elements if time permits, such as a self-cleaning option and / or an attractant / insecticide dispensing system.
- Process collected data and evaluate it.
- Complete write-up of dissertation.
- Prepare presentation.

A Gantt chart of the work is included in Figure 27, taking account of the proposed workflow.



	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		Literature review	1 mon	Mon 16/03/20	Fri 10/04/20	
2		Focus research question and finalise objectives	1 wk	Mon 13/04/20	Fri 17/04/20	1
3		Literature review and technical learning	6 wks	Mon 20/04/20	Fri 29/05/20	2
4		**Progress report**	1 day	Wed 27/05/20	Wed 27/05/20	
5		**Develop prototype**	**49 days**	**Thu 28/05/20**	**Tue 4/08/20**	3
6		Download software and obtain hardware, traps and consumables	1 day	Thu 28/05/20	Thu 28/05/20	4
7		Set up the Raspberry Pi & ancilliary hardware and begin programming	2 days	Mon 8/06/20	Tue 9/06/20	4FS+7 days
8		Import images and develop algorithm to optimise images for classification	2 wks	Wed 10/06/20	Tue 23/06/20	7
9		Train deep learning and classification algorithm with available images	2 wks	Wed 24/06/20	Tue 7/07/20	8
10		Trap fruit flies and photograph them in traps	8 wks	Mon 8/06/20	Fri 31/07/20	4FS+7 days
11		Obtain images of fruit flies from additional sources and process them for model training	8 wks	Mon 8/06/20	Fri 31/07/20	4FS+7 days
12		Apply trained model to captured images	4 wks	Wed 8/07/20	Tue 4/08/20	9
13		Develop algorithm to export events and weather data for forwarding to recipient	2 wks	Wed 10/06/20	Tue 23/06/20	7
14		Process and evaluate experimental data	3 wks	Wed 5/08/20	Tue 25/08/20	5
15		Write up dissertation	2 wks	Wed 26/08/20	Tue 8/09/20	14,13
16		**Draft dissertation**	0 days	Wed 9/09/20	Wed 9/09/20	15
17		Review	2 wks	Wed 9/09/20	Tue 22/09/20	16
18		Finalise dissertation and prepare presentation	2 wks	Wed 23/09/20	Tue 6/10/20	17
19		**Hand in dissertation**	0 days	Thu 15/10/20	Thu 15/10/20	18

Figure 27 Gantt chart of project schedule and workflow.

B.2
RISK ASSESSMENT AND HEALTH, SAFETY AND ENVIRONMENT CONSIDERATIONS

The risk matrix in Table 7 was to used evaluate the potential risk of the project activities. An unmitigated hazard assessment is included in Table 8. This matrix evaluates hazards without taking specific precautions. An evaluation of hazard levels after implementation of specific mitigation measures is included in Table 9, along with proposed mitigation measures.

Table 7 Risk assessment matrix to evaluate project risk.

		Consequence				
		Minor equipment damage	Major equipment damage	Minor injury or illness or environmental impact	Major injury or illness or environmental impact	Possible death or environmental catastrophe
Likelihood	Very rarely (30 years)	1	2	3	5	7
	Rarely (10 years)	2	4	6	10	14
	Occasionally (2 years)	3	6	9	15	21
	Regularly (1 year)	4	8	12	20	28
	Frequently (1 Month)	5	10	15	25	35
	Continuously	6	12	18	30	42
		Low risk	High risk	Very high risk	Extreme risk	

Table 8 Assessment of unmitigated hazards.

Hazard No	Hazard	Likelihood /consequence	Risk
1	Electrical failure or malfunction.	3 x 5	15
2	Eye injury through working in trees.	4 x 5	20
3	Accident during travel to site.	2 x 7	14
4	Sunburn	4 x 3	12

5	Toxicity from handling of pheromone and insecticide (Malathion) lures.	3 x 7	21
6	Exposure of third parties to insecticide and pheromone chemicals at home.	2 x 7	14
7	Exposure of third parties to insecticide and pheromone chemicals at field sites.	2 x 7	14
8	Environmental impacts from the use of pheromone or bait & insecticide chemicals.	3 x 3	9

Table 9 Assessment of hazards after mitigtation.

Hazard No	Hazard mitigation measures	Likelihood /consequence	Risk
1	Electrical malfunction is not considered a high risk but can be reduce through good design and workmanship. Use of safety glasses and appropriate work tools can avoid major injuries if explosive failure should occur during the testing process.	2 x 3	6
2	Risk of eye injury can be reduced through the use of safety glasses or goggles.	1 x 4	4
3	Avoid unnecessary travel, especially at night and during dawn and dusk. Obey road rules and avoid driving when tired.	1 x 7	7
4	Wear a hat, sunglasses (dark safety glasses) and long clothes and use sunscreen (with re-application).	2 x 3	6
5	Wear rubber gloves when handling insecticide / lure chemicals to avoid skin contact. Maintain awareness not to touch skin or eyes elsewhere and have a telephone available to phone emergency services. Handle chemicals outdoors and wear a breathing mask, rated for removal of organic vapours, to avoid breathing fumes.	1 x 5	5
6	Handle, store and dispose of chemicals according to their MSDSs and use them according to guidelines in the backyard where there is restricted access.	1 x 5	5

7	Do not leave unused chemicals on site, use lures according to guidelines and hang traps away from locations that are likely to draw attention to them.	1 x 5	5
8	Ensure that adequate equipment is available to dispense chemicals without spillages and remove spent materials from site.	1 x 3	

Based on the risk assessment, the project was not considered to pose any significant risks as long as basic safety precautions were taken. For handling of traps and chemicals this included use of personal protective equipment (PPE), particularly gloves and safety glasses. Gloves would also to be used for handling insects, to avoid contact with potential pathogens. For electronic work the use of appropriate tools and safety glasses was specified to prevent eye injuries and electric burns and shocks.

If traps were to be established at field sites away from home, they would preferably be hung in locations that would not draw attention to them and / or where there is adequate access control. This would assist in preventing third party contact and theft.

Proper handling of chemicals and disposal of containers and contaminated PPE would limit any environmental risk. Traps were to be removed from their locations after testing, unless a decision were made to continue using them.

Travel risk was not be applicable, as the project was carried out at home. Limited opportunities for mitigation would be possible otherwise, aside from well-planned and responsible travel.

Appendix C

Software

Note: All applicable file names are listed within the algorithms.

C.1
Main control algorithm

```
# File: MAINemail.py
# Date: 21 September 2020
# Description: Main control algorithm for automated fruit fly trap.
# Author: P. Rousseau.
# Modifications: None.

# import sys
# sys.path.append('../')
# sys.path.append('/python-bme280-master')
# sys.path.append('../MIPI_Camera/RPI')
# sys.path.append('../MIPI_Camera/RPI/ISP')
import subprocess
#import os
import time
import csv

#Run the TPH sensor code and write csv file for TPH readings
TPH = subprocess.run(["sudo" , "python3" , "python-bme280-master/TPHtcsv.py"],
check=True)

subprocess.run(["sudo" , "python3" , "../MIPI_Camera/RPI/flash_on.py"]) #Switch on
neopixel illumination
time.sleep(6)

#subprocess.run(["sudo" , "python3" , "../MIPI_Camera/RPI/ISP/CAMERA.py"]) #Activate
image collection and processing. Disabled.

subprocess.run(["sudo" , "python3" , "../MIPI_Camera/RPI/ISP/flash_off.py"]) #Activate
image collection and processing.

subprocess.run(["sudo" , "python3" , "Blob_detector_CV2_calibrated.py"]) #Activate image
collection and processing.

time.sleep(1)

fields: list = []
row: list = []

with open('bme280out.csv', 'r') as csvTPH:
    TPHfile = csv.reader(csvTPH)
    TPHlist = list(TPHfile)
    #print(TPHlist)
with open('flycount.csv', 'r') as csvfly:
    flyfile = csv.reader(csvfly)
    flylist= list(flyfile)
    #print(flylist)

with open('output.csv', 'w') as outfile:
    output = csv.writer(outfile)

    fields = TPHlist[0] + flylist[0]
```

```
    output.writerow(fields)

    for item in range(1,len(TPHlist)):
        row = TPHlist[item] + flylist[item]
        output.writerow(row)

subprocess.run('mpack -s "Fly trap report" output.csv email@domain.com', shell=True,
check=True)
```

C.2
Bosch BME280 TPH sensor control algorithm

```
# File: TPHtcsv_multi.py
# Date: 21 September 2020
# Description: Control algorithm for Bosch BME280 temperature, pressure, humidity sensor.
#              Records clock time and, after a set delay (default 30 seconds) a set number
#              of separate measurements (default 15) is averaged and recorded.
#              Results are saved to a csv file with default name "bme280out.csv".
# Author: P. Rousseau, all new code except for start code by C. Nicolai, as included below.
# Modifications: All except for start code provided.

#Start Demo Code
#--------------------------------------------------------------------------------
#!/usr/bin/env python

#import bme280

#def main():
#    bme = bme280.Bme280()
#    bme.set_mode(bme280.MODE_FORCED)
#    t, p, h = bme.get_data()
#    print("Temperature: %f °C" % t)
#    print("Pressure: %f P" % p)
#    print("Humidity: %f %%" % h)

#if __name__ == '__main__':
#    main()
#--------------------------------------------------------------------------------
#End Demo Code

import bme280
import time #To add delay and time of day.
import csv

interrupt = False
intervals = 15 #Number of measurements to average.

delay = 30 #Delay between measurements in seconds.

#CSV code examples from https://www.geeksforgeeks.org/writing-csv-files-in-python/ used
to develop csv output.
fields = ['Local time' , 'Temperature C' , 'Pressure Pa' , 'Rel-humidity %']
filenm = "bme280out.csv" #Name of csv file

def main():
    bme = bme280.Bme280()
    bme.set_mode(bme280.MODE_FORCED)
    t, p, h = bme.get_data()
#     print(f'Temperature: {t}')
#     print(f'Pressure: {p}')
#     print(f'Humidity: {h}')
    return [t, p, h]
```

```
if __name__ == '__main__':

#Create csv file with headings and first row of data.
    tT, pT, hT = 0.0, 0.0, 0.0 #Reinitialise totals
    #time.sleep(delay)
    for i in range(intervals): #To obtain average of specified number of measurements. PR
        t, p, h = main()
        #time.sleep(0.3)   #Delay to reduce data flow
        tT += t
        pT += p
        hT += h
        #print(tT, pT, hT)
        if i == intervals-1: #Calculate average of summed intervals
            t = tT/intervals
            p = pT/intervals
            h = hT/intervals
            loctime = time.localtime() #Based on python time module reference
            loctime = time.asctime(loctime)
            print(f'Local time {loctime}, Temp {round(t,2)} C, Pres {round(p,0)} Pa, Hum
{round(h,2)}%')
            vals = [loctime,t, p, h]
            with open(filenm, 'w') as csvfile: #Write averaged values to csv file
                csvwriter = csv.writer(csvfile)
                csvwriter.writerow(fields)
                csvwriter.writerow(vals) #Function appends rows instead of writing only one.

interrupt = False
#Append csv file with new data.
while interrupt == False:
    try:
        tT, pT, hT = 0.0, 0.0, 0.0 #Reinitialise totals
        time.sleep(delay)
        for i in range(intervals): #To obtain average of specified number of measurements. PR
            t, p, h = main()
            #time.sleep(0.3)   #Delay to reduce data flow
            tT += t
            pT += p
            hT += h
            #print(tT, pT, hT)
            if i == intervals-1: #Calculate average of summed intervals
                t = tT/intervals
                p = pT/intervals
                h = hT/intervals
                loctime = time.localtime() #Based on python time module reference
                loctime = time.asctime(loctime)
                print(f'Local time {loctime}, Temp {round(t,2)} C, Pres {round(p,0)} Pa, Hum
{round(h,2)}%')
                vals = [loctime,t, p, h]
                with open(filenm, 'a', newline = '') as csvfile: #Write averaged values to csv file
                    csvwriter = csv.writer(csvfile)
                    #csvwriter.writerow(fields)
                    csvwriter.writerow(vals) #Function appends rows instead of writing only one.
                    #csvwriter.writerows(rows) #To be implemented if multiple readings are
recorded
    except KeyboardInterrupt: #Escapes and saves file using "Ctrl + C".
```

```
        interrupt = True
print('\nFile saved')

#    t , p , h = [tT, pT, hT]/intervals
```

C.3
Algorithm to switch on neopixel lighting

```
# File: flash_on.py
# Date: 07 September 2020
# Description: Switch on Adafruit neopixel 12 LED ring with white light at 0.5 brightness.
# Author: P. Rousseau.
# Modifications: Modified from code provided at https://learn.adafruit.com/neopixels-on-raspberry-pi/python-usage (web page version 12 Sept 2014)

# Simple test for NeoPixels on Raspberry Pi
import time
import board
import neopixel

# Choose an open pin connected to the Data In of the NeoPixel strip, i.e. board.D18
# NeoPixels must be connected to D10, D12, D18 or D21 to work.
pixel_pin = board.D18

# The number of NeoPixels
num_pixels = 12

# The order of the pixel colors - RGB or GRB. Some NeoPixels have red and green reversed!
# For RGBW NeoPixels, simply change the ORDER to RGBW or GRBW.
ORDER = neopixel.RGBW

pixels = neopixel.NeoPixel(
    pixel_pin, num_pixels, brightness=0.5, auto_write=False, pixel_order=ORDER
)
while True:
    # Comment this line out if you have RGBW/GRBW NeoPixels
    #pixels.fill((255, 0, 0))
    # Uncomment this line if you have RGBW/GRBW NeoPixels
    pixels.fill((0, 0, 0, 255))
    pixels.show()

    break

#     rainbow_cycle(0.001)  # rainbow cycle with 1ms delay per step
```

C.4
Algorithm to switch off neopixel lighting

```
# File: flash_off.py
# Date: 07 September 2020
# Description: Switch off Adafruit neopixel 12 LED ring.
# Author: P. Rousseau.
# Modifications: Modified from code provided at https://learn.adafruit.com/neopixels-on-raspberry-pi/python-usage (web page version 12 Sept 2014)

# Simple test for NeoPixels on Raspberry Pi
import time
import board
import neopixel

# Choose an open pin connected to the Data In of the NeoPixel strip, i.e. board.D18
# NeoPixels must be connected to D10, D12, D18 or D21 to work.
pixel_pin = board.D18

# The number of NeoPixels
num_pixels = 12

# The order of the pixel colors - RGB or GRB. Some NeoPixels have red and green reversed!
# For RGBW NeoPixels, simply change the ORDER to RGBW or GRBW.
ORDER = neopixel.RGBW

pixels = neopixel.NeoPixel(
    pixel_pin, num_pixels, brightness=0.2, auto_write=False, pixel_order=ORDER
)

#
while True:

    pixels.fill((0, 0, 0, 0))
    pixels.show()

    break

#     rainbow_cycle(0.001)  # rainbow cycle with 1ms delay per step
```

C.5
Modified Camera Control Algorithm

```
# File: CAMERA.py
# Date: 7 September 2020
# Description: Control algoritm used for Arducam camera.
# Author: P. Rousseau.
# Modifications: Modified version of preview.py demo file, located in the RPI/ISP folder of the MIPI_camera installation folder.
#                MIPI software version installed on 29 June 2020 according to https://github.com/ArduCAM/MIPI_Camera/tree/master/RPI


import sys
sys.path.append('../')
import arducam_mipicamera as arducam
import v4l2 #sudo pip install v4l2
import time
import numpy as np
import cv2
import os
from isp_lib import *

os.system("sudo python3 flash_on.py")

def set_controls(camera):
    try:
        print("Reset the focus...")
        camera.reset_control(v4l2.V4L2_CID_FOCUS_AUTO)                                  # (v4l2.V4L2_CID_FOCUS_ABSOLUTE) changed by P Rousseau
        time.sleep(6) #Added to improve autofocus P Rousseau
    except Exception as e:
        print(e)
        print("The camera may not support this control.")  #This error is printed but it appears that Autofocus is working. P Rousseau

    try:
        print("Enable Auto Exposure...")
        camera.software_auto_exposure(enable = True)
        print("Enable Auto White Balance...")
        camera.software_auto_white_balance(enable = True)
    except Exception as e:
         print(e)

def resize(frame, dst_width=1000): #Changed from 640 P Rousseau
    height = frame.shape[0]
    width = frame.shape[1]

    scale = (dst_width * 1.0) / width
    return cv2.resize(frame, (int(scale * width), int(scale * height)))


if __name__ == "__main__":
    try:
```

```
        camera = arducam.mipi_camera()
        print("Open camera...")
        camera.init_camera()
        _isp = isp(camera.camera_instance)
        print("Setting the mode...")
        camera.set_mode(0)
        fmt = camera.get_format()
        fmt = (fmt["width"], fmt["height"])
        print("Current resolution is {}".format(fmt))
        set_controls(camera)

        start_time = time.time()
        do_change = True
        while True:
            data = camera.capture(encoding = 'raw')
            # Use different variable names to avoid memory being released
            frame = arducam.unpack_raw10_to_raw8(data.buffer_ptr, fmt[0], fmt[1])
            frame = cv2.cvtColor(frame.as_array.reshape((fmt[1], fmt[0])),
cv2.COLOR_BAYER_RG2BGR)
            _isp.run_awb(frame)
            _isp.run_ae(frame)

            disp = resize(frame)

            cv2.imshow("Arducam", disp)
            ret = cv2.waitKey(10)
            if ret == ord('q'):
                break

            if time.time() - start_time >= 5 and do_change:
                do_change = False
                camera.set_mode(3)  #Changed to Mode 3 for high resolution image PRousseau
                fmt = camera.get_format()
                fmt = (fmt["width"], fmt["height"])
                start_time = time.time()
            if time.time() - start_time >= 5 and not do_change:
                cv2.imwrite("Test.jpg", frame) #Need to add a routine to check for max number of
files (to free storage memory) and auto increment number PRousseau
                time.sleep(3) #Delay for preview inserted by P Rousseau
                break

        print("Close camera...")  #Moved from below (see comment) P Rousseau
        camera.close_camera()

        #Enter code for OpenCV calculations here. P Rousseau

        # Release memory
        del frame
        del data
        #Move close camera above release memory from here P Rousseau
    except Exception as e:
        print(e)
time.sleep(1)
os.system("sudo python3 flash_off.py")
```

C.6
Simple Blob Detector input file - calibrated

```
# File: Blob_detector_CV2_calibrated.py
# Date: 18 July 2020 installed version of OpenCV
# Description: Calibrated OpenCV Simple Blob Detector input file.
# Author: P. Rousseau using code examples from references listed below.
# Modifications: Included and updated adjustable parameters as necessary. Implemented export function for number of keypoints counted (assumed to be flies).

#!/usr/bin/env python
# coding: utf-8

# In[21]:

#!/usr/bin/python
#Code modified from https://stackoverflow.com/questions/8076889/how-to-use-opencv-simpleblobdetector

# Standard imports
import cv2
import numpy as np
import csv #CSV code exemples from https://geeksforgeeks.org/writing-csv-files-in-python/

field = ['Number of flies']

# Read image
im = cv2.imread("images/camera_output.jpg", cv2.IMREAD_GRAYSCALE) #The ouput image file generated by the camera should be referenced here.
#im = cv2.equalizeHist(im) #Equalize histogram to improve dynamic range.
#im = cv2.blur(im, ksize = (40,40))
# Setup SimpleBlobDetector parameters.
params = cv2.SimpleBlobDetector_Params()

# Change thresholds
params.thresholdStep = 1
params.minThreshold = 40
params.maxThreshold = 160

# Filter by colour (reported as broken in https://www.learnopencv.com/blob-detection-using-opencv-python-c/)
#params.filterByColor = True
#params.blobColor = 100

# Filter by Area.
params.filterByArea = True
params.minArea = 4000

# Filter by Circularity
params.filterByCircularity = True
params.minCircularity = 0.1
params.maxCircularity = 0.3

# Filter by Convexity
```

```
params.filterByConvexity = True
params.minConvexity = 0.4
#params.maxConvexity = 0.8

# Filter by Inertia
params.filterByInertia = True
params.minInertiaRatio = 0.1

# Create a detector with the parameters
detector = cv2.SimpleBlobDetector_create(params)

# Detect blobs.
keypoints = detector.detect(im)

# Draw detected blobs as red circles.
# cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS ensures
# the size of the circle corresponds to the size of blob

im_with_keypoints = cv2.drawKeypoints(im, keypoints, np.array([]), (0,0,255), cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

# Show blobs
#cv2.imshow("Keypoints", im_with_keypoints)
#cv2.waitKey(1)

cv2.imwrite('fly_detect.png', im_with_keypoints)
#print(f'{keypoints}')
print(f'The number of keypoints is {len(keypoints)}')

val = [len(keypoints)]

with open('flycount.csv', 'w') as csvfile: #Based on https://www.geeksforgeeks.org/writing-csv-files-in-python/
    csvwriter = csv.writer(csvfile)
    csvwriter.writerow(field)
    csvwriter.writerow(val)
    #Use csvwriter.writerow(rows) to store multiple readings.
```

C.7
Feature extraction and identification algorithm

```
# File: feature_extraction_and_SVM_py3.py. Original filename
feature_extraction_and_SVM.py
# Date: 31 August 2020
# Description: Modified version of insect identification algorithm for Python 3. Runs but has
not been checked for correct operation.
#              Original data downloaded from: https://github.com/valanm/off-the-shelf-insect-
identification
# Author: Valan et al. 2019 modified by P. Rousseau.
# Modifications: Processed through Python 2to3 and manually updated as required.

import tensorflow as tf
from tensorflow import keras  #Added PRousseau

# import tensorflow.compat.v1 as tf   #Added PRousseau according to
https://www.tensorflow.org/guide/migrate
# tf.disable_v2_behavior()

import matplotlib.pyplot as plt

import os
import numpy as np

from sklearn.svm import SVC, LinearSVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import StratifiedKFold

from tensorflow.keras.models import Sequential, Model                    #Added tf as
keras is now part of tensorflow
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import GlobalAveragePooling2D, GlobalMaxPooling2D
from tensorflow.keras.applications.vgg16 import VGG16

def save_all_features(nb_samples, source="images", dest="features", input_size = (416,
416), batch_size=6):

    """
    This function extracts features after every MaxPool layer in VGG16.
        Input:
            - nb_samples - total number of images you have
            - source     - directory whit images as shown below:
                        - images:
                            -class 0:
                                - im 0
                                - im 1
                                - im 2...
                            -class 1:
                                - im 0
                                - im 1...,
            - dest       - save features to this directory
            - input_size - image size in pixels
```

```
        - batch_size - number of images per epoch (larger input_size requires smaller
batches)

    Output:
        saves to "dest" directory:
            - X - features
            - Y - labels
            - filenames
    """

    # check if the directory exists, and if not make it
    if not os.path.exists(dest):
        os.makedirs(dest)

    # define image height and width
    (img_height, img_width) = input_size

    # build the VGG16 network and extract features after every MaxPool layer
    model = VGG16(weights='imagenet', include_top=False)

    c1 = model.layers[-16].output
    c1 = GlobalAveragePooling2D()(c1)

    c2 = model.layers[-13].output
    c2 = GlobalAveragePooling2D()(c2)

    c3 = model.layers[-9].output
    c3 = GlobalAveragePooling2D()(c3)

    c4 = model.layers[-5].output
    c4 = GlobalAveragePooling2D()(c4)

    c5 = model.layers[-1].output
    c5 = GlobalAveragePooling2D()(c5)

    model = Model(inputs=model.input, outputs=(c1,c2,c3,c4,c5))

    # define image generator without augmentation
    datagen = ImageDataGenerator(rescale=1./255.)
    generator = datagen.flow_from_directory(
            source,
            target_size=(img_height, img_width),
            batch_size=batch_size,
            class_mode="sparse",
            shuffle=False)

    # generate and save features, labels and respective filenames
    steps = nb_samples/batch_size+1
    X = model.predict_generator(generator,steps)
    Y = np.concatenate([generator.next()[1] for i in range(0, generator.samples, batch_size)])
    names = generator.filenames

    for n, i in enumerate(X):
        with open(dest+"X-"+str(img_height)+"-c"+str(n+1)+"-AVG.npy", 'w') as f:
                np.save(f, i)
```

```
    if not os.path.exists(dest+"Y.npy"):
        with open(dest+"Y.npy"  , 'w') as f:
            np.save(f, Y)
    if not os.path.exists(dest+"filenames.npy"):
        with open(dest+"filenames.npy"  , 'w') as f:
            np.save(f, names)

def kfoldSVM_on_features(X, Y):
    # define 10-fold cross validation test harness
    kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=555)
    cvscores, splits = [],[]
    for train, test in kfold.split(X, Y):
        clf = LinearSVC(C=1.0, loss='squared_hinge', penalty='l2',multi_class='ovr')
        clf.fit(X[train], Y[train])
        y_pred = clf.predict(X[test])
        acc = accuracy_score(Y[test],y_pred)*100
        cvscores.append(acc)
        splits.append((Y[test], y_pred))
    print(("Accuracy score averaged across 10 kfolds %.2f%% (+/- %.2f%%)" %
(np.mean(cvscores), np.std(cvscores))))


def evaluate(dest="features", size=416, strategy = "-AVG"):
    #
    size = str(size)
    l1 = np.load(dest+"X-"+size+"-c1"+strategy+".npy")
    l2 = np.load(dest+"X-"+size+"-c2"+strategy+".npy")
    l3 = np.load(dest+"X-"+size+"-c3"+strategy+".npy")
    l4 = np.load(dest+"X-"+size+"-c4"+strategy+".npy")
    l5 = np.load(dest+"X-"+size+"-c5"+strategy+".npy")
    a_all = np.concatenate([l1,l2,l3,l4,l5], 1)

    X =[l1, l2, l3, l4, l5, a_all]

    Y = np.load(dest +"Y.npy")

    for n, x in enumerate(X):
        print()
        if n==5:
            print("fused features across all conv blocks")
        else:
            print("conv block", n+1)
        print("without normalization")
        kfoldSVM_on_features(x, Y)
        print("with square root normalization")
        x = np.sqrt(np.abs(x)) * np.sign(x)
        kfoldSVM_on_features(x, Y)

input_size = (416,416)
nb_samples = 240
save_all_features(nb_samples, source="./images/", dest="./features/",
input_size=input_size)

print()
print()
```

```
print("evaluating dataset with input size", input_size, "and GlobalAveragePooling2D")
evaluate(dest="./features/", size=input_size[0])
```