Ollscoil
Teicneolaíochta
an Atlantaigh

Atlantic
Technological
University

SmartCycle

Project Engineering

Year 4

Ciara Crowe

Bachelor of Engineering (Honours) in Software and Electronic Engineering

Atlantic Technological University

2024/2025

Declaration

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering (Honours) in Software and Electronic Engineering at Atlantic Technologic University Galway.

This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

____

Acknowledgements

I would like to thank my supervisor Niall O’Keefe for his guidance throughout my final year project.

I would also like to thank all of the lectures who worked with us for our project.

Table of Contents

1 Summary

SmartCycle is a smart bin system that detects different objects using computer vision. The objects are classified and sorted into the detected bin. The bin features two servo motors which physically opens the bin while the addition of an infrared sensor notifies the user if the bin needs emptying. Using a temperature and humidity sensor will help keep the compost conditions optimal. The combination of hardware and computer based classification creates an intelligent automated waste sorting system.

With the help of a waste collection database, I will train and deploy my machine learning model from Edge Impulse onto the Raspberry Pi. Using a camera, the platform predicts the objects classification and its position, drawing a box around the tested object. If the predicated classification is detected, the Raspberry Pi handles the rotation of the servo motor, which acts as a door, opening and closing the appropriate bin. The DHT11 sensor collects real time data every minute while the infrared sensor is active to check if the bin is full. The data is collected and stored in a database which is displayed in a webserver I have created. In addition to this, the user will be able to see the live feed to check what is being predicated by the model.

This project features both software and hardware components to make real time predictions on models based on the trained data. The computer vision detection feature aids people to make the right choice when sorting waste by identifying the objects class, reducing the chance of human error.

2 Poster

BEng (Honours) Software & Electronic Engineering

SmartCycle

By Ciara Crowe

Ollscoil Teicneolaíochta an Atlantaigh

Atlantic Technological University

Introduction:

SmartCycle is a smart waste management system which uses computer vision to detect waste objects. It sorts them into two classes, Recycling or Compost. Using servo motors, the bin is opened based on what class is detected.

To ensure my bin is user friendly, I have added an infrared sensor which lets the user know if the bin is full. While the addition of my DHT11 temperature and humidity sensor ensure that optimal composting conditions are taking place.

All data is containerised via Docker, saved in my TinyDB database and displayed in a web page. This presents the real time data on the waste collected and the compost readings.

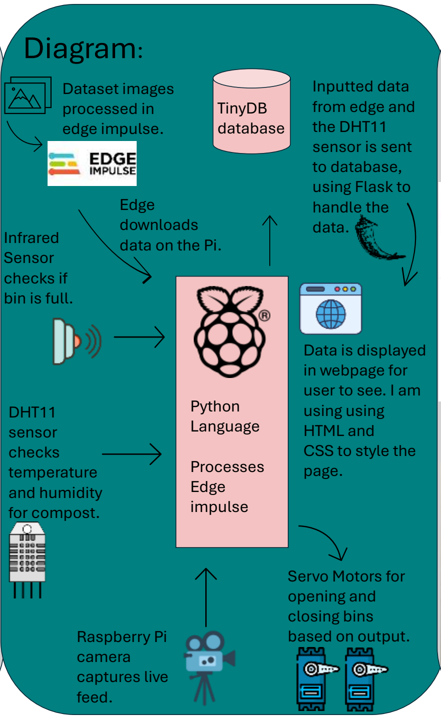
Hardware:

- Raspberry Pi – Microcontroller
- DHT11 – Temperature and humidity sensor for composting.
- Infrared Senor – Checks if bin is full.

Software:

- Edge Impulse – Generates my machine learning model.
- Python – Controls servo motors, sensor and processes data.
- TinyDB – NoSQL database which stores data in JSON.
- HTML and CSS: Structuring and designing the webpage.

Diagram:



Dataset images processed in edge impulse.

TinyDB database

Inputted data from edge and the DHT11 sensor is sent to database, using Flask to handle the data.

Edge downloads data on the Pi.

Infrared Sensor checks if bin is full.

Python Language

Processes Edge impulse

Data is displayed in webpage for user to see. I am using using HTML and CSS to style the page.

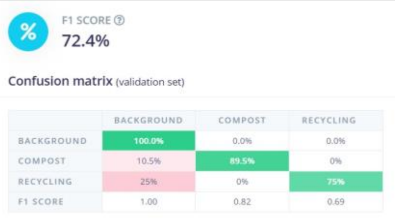
DHT11 sensor checks temperature and humidity for compost.

Servo Motors for opening and closing bins based on output.

Raspberry Pi camera captures live feed.

Results:

This was the output score from edge impulse. It received an overall F1 score of 72.4%, with Compost being accurately identified at 89.5% and Recycling at 75%.



F1 SCORE

72.4%

Confusion matrix (validation set)

	BACKGROUND	COMPOST	RECYCLING
BACKGROUND	100.0%	0.0%	0.0%
COMPOST	10.5%	89.5%	0%
RECYCLING	25%	0%	75%
F1 SCORE	1.00	0.82	0.69

These are the live images which show the accuracy classification labels.

Recycling = 0.77

Compost = 0.95

3 Introduction

‘Artificial intelligence is not a substitute for human intelligence; it is a tool to amplify human creativity and ingenuity’[1]. Artificial Intelligence (AI) is a topic which has interested me for a number of years. When considering ideas for my final year project, I decided to focused on AI. A big topic we see repeatedly in our society is our waste management control. According to an audit, “ Only 53% can accurately identify what can actually be recycled”[2]. Knowing the difference of what can and can’t be recycled can have a huge impact to society. The environmental advantages of having a good recycling management in place include the reduction of landfills and conserving natural resources for those by reducing the need to collect new raw materials[3]. This gave me an idea for a project. By using AI, I could train a waste dataset to detect images and sort them into the correct bins, aiding people to make the right choices. As this topic focused on environmental issues, I set out a challenge to see if I could produce a low impact sustainable product.

The report will guide you through the challenges I faced along the way and how I overcame them and successfully created my SmartCycle system.

4 Background

Before starting my project, it was important that I carried out research in order to have a better understanding of the task I faced ahead. This knowledge helped shape the process of my project and how I approached different tasks. I will talk about the background research in the following sub sections.

4.1 Dataset

In order to train my model I needed a dataset. This dataset needed to contain the right data so I could sort waste images into the appropriate classes. For my project, I decided I was going to have two classes, recycling and compost. I needed to find a dataset that had data for both of these classes, which ended up being quiet challenging. For the first couple of months I used Taco Dataset[4], however I experienced a few problems with this. This had a numerous amount of data for recycling but not so much for compost. Compost was also broken into smaller classes like fruit and vegetables, bio, eggshells etc. As my project was small, I wanted to use specific classes like fruit and vegetables for my compost class. As this wasn't a main class there wasn't enough data for this. This meant that when I tried to train the data, there was not enough images for compost which resulted in the compost class not being detected. The model struggled to train the compost as it was under sampled compared to the recycling class. To combat this issue, I decided to do more research and find a different dataset which would be more appropriate to my project. I changed my dataset to CompostNet[5], which had the same number of data for each class I was using. When I changed my dataset, my compost class went from a 0% accuracy rate to 89.5% accuracy rate. This helped me understand that researching the dataset before starting your project can save a lot of time. Making sure you have enough data for each class is crucial for preventing undesired outcomes. Below is data from the dataset which I have trained

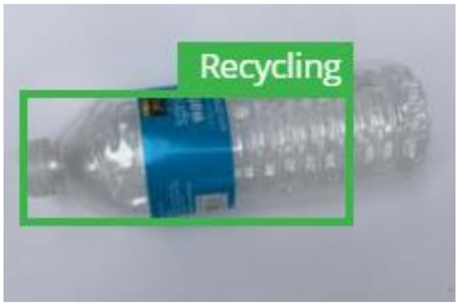


Figure 4.1-1 Recycling

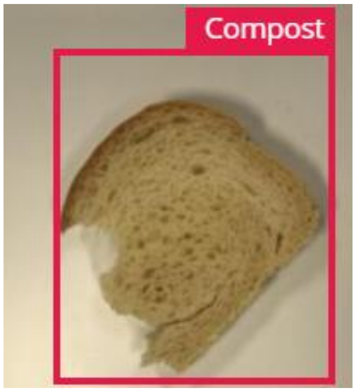


Figure 4.1-2 Compost

4.2 Raspberry Pi and Python

It was important to me that I incorporated hardware into my project so I decided to use a microcontroller. I wanted to use a cost effective but powerful chip. Deciding on what language to use was important as this would be used in conjunction with my microcontroller[6]. After much consideration I decided to use Python as this language is very popular with computer vision and I had just worked with Python for nine months on my work placement. To implement a Python computer based project the Raspberry Pi was the best option for me to use. In order to have strong processing power it was recommended that I chose a model four or later. I chose to work with the Raspberry Pi 4B.

After receiving my Raspberry Pi I needed to download the Operating System (OS) which was an easy enough process to follow[7]. Using a 32GB micro SD card which was recommended by the Raspberry Pi documentation, I used Raspberry Pi Imager to download the newest version of Raspberry Pi’s OS, Debian Bookworm, from my laptop onto the micro SD card. Once this was downloaded I inserted the SD card into the Raspberry Pi. Using a 15W C power supply, a mouse, a keyboard, a micro HDMI cable and a monitor I was able to power on my Raspberry Pi. Now that I had my Raspberry OS set up, I could begin working on my project.

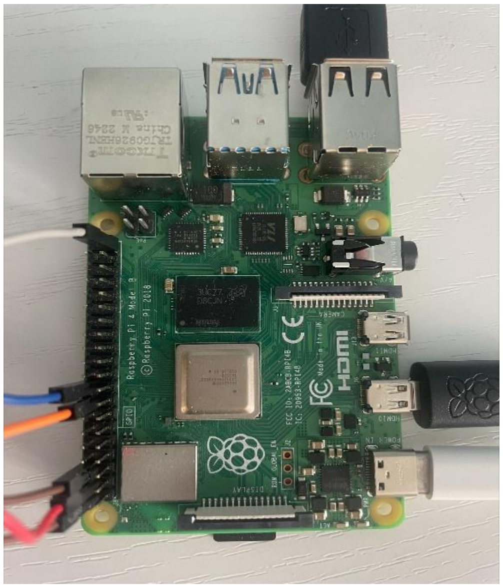


Figure 4.2-1 Raspberry Pi

5 Project Architecture

This diagram will help show how my project works visually before getting into the details.

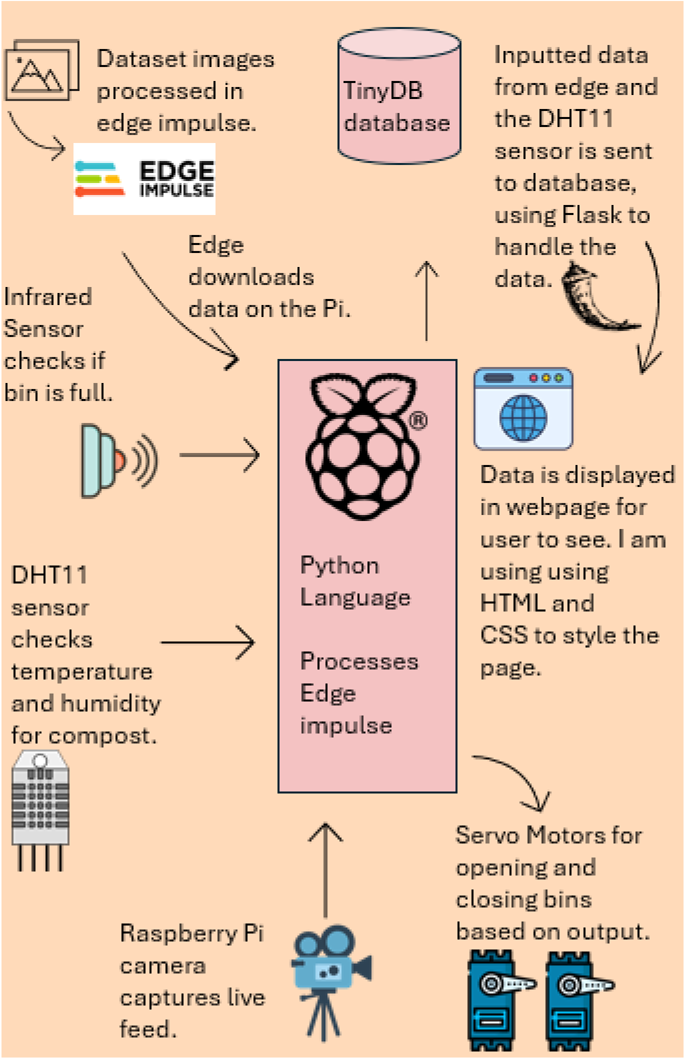
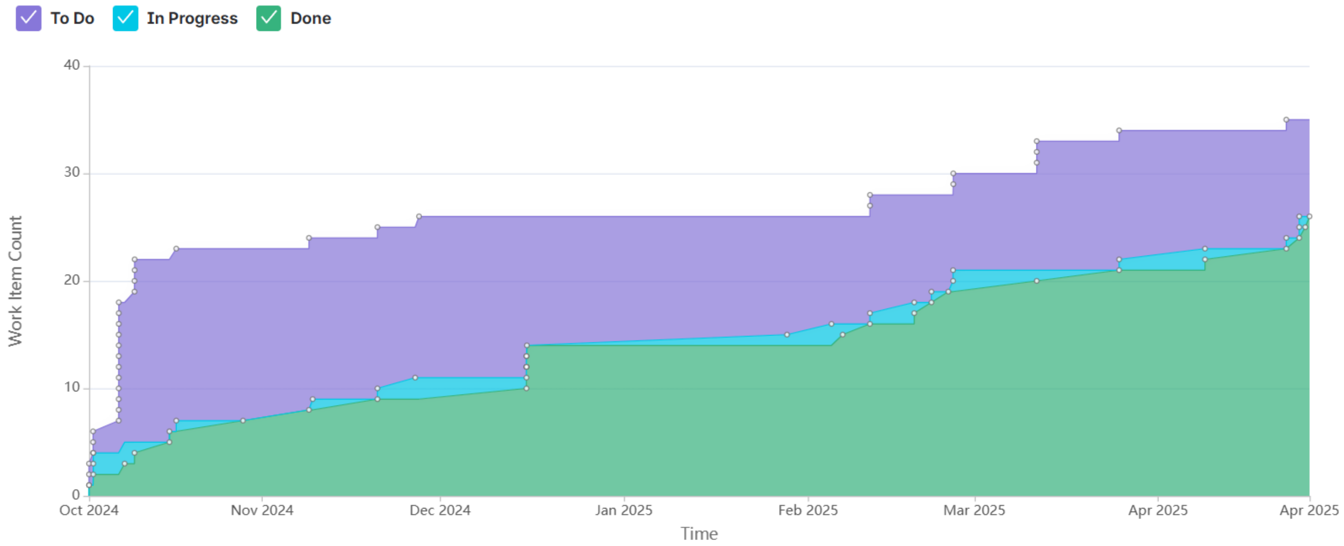

Dataset images processed in edge impulse.
EDGE IMPULSE
TinyDB database
Inputted data from edge and the DHT11 sensor is sent to database, using Flask to handle the data.
Edge downloads data on the Pi.
Infrared Sensor checks if bin is full.
Python Language
Processes Edge impulse
Data is displayed in webpage for user to see. I am using using HTML and CSS to style the page.
DHT11 sensor checks temperature and humidity for compost.
Servo Motors for opening and closing bins based on output.
Raspberry Pi camera captures live feed.

Figure 5-1 Architecture Diagram

6 Project Plan

I used Jira to manage my project throughout my final year. Before starting my project, I created a number of tickets and sorted them into epics. I created in total eight sprints throughout the semester. These sprints typically ran for two weeks or more depending if I had reached my goal set out at the start of the sprint. If I finished a ticket earlier than expected I dragged in tickets from my backlog. Overall, having a Jira management system helped me keep track of where I was and allowed me to get an estimate of when I could finish my project. This is a picture of the flow of my sprint over the eight months. From October to January I had a few set backs and it took me a while to maintain a steady workflow. From February onwards when my project started coming together.



7 Components and Technologies Used

Throughout this section I will talk about each of the hardware, software, and applications I have used during the development of my project. For each method I will include the setup, the role it plays in my project and any issues or problems I experienced and how I handled it.

7.1 Hardware

This section includes the hardware which I have used. The following subheadings will dive into more detail for my motors, infrared sensor, DHT11 sensor and the Raspberry Pi.

7.1.1 Motors

One of my main goals for my project was to act autonomously. To incorporate this idea, my plan was to have the lids of the bin open and close independently. A servo motor was the best choice for this as I could set the lid to open and close at a specific angle. I also wanted to use a motor which uses low power. The servo motor was the best choice as it only uses power when rotating. By adding two servo motors to my project I could open and close the bin based on what class was detected by my model. Once a detection is made the motor will turn 90 degrees to open followed by a delay to allow the user to place the object in the bin. The motor will then turn back to its original position. I created the following image on KiCad EDA which shows the motors outputs.

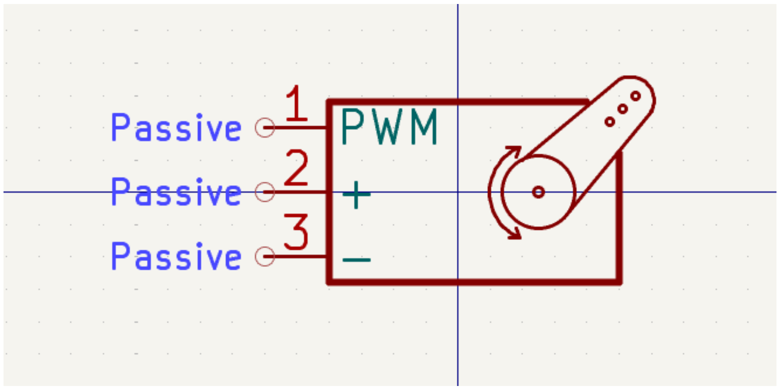


Figure 7.1.1-1 Servo Motor

The servo have 3 outputs; Ground(-), Power(+) and PWM pin. I connected these directly to the Raspberry Pi, power pins, 5V, GRND and to GPIO 23[8]. I used PWM to control the speed of my motors by importing the library GPIO. The following piece if code is an example of how my motors work using PWM when the motors move.

```
def open_bin(pwm):

    print("Opening bin")

    pwm.ChangeDutyCycle(5.0)

    time.sleep(6)

    pwm.ChangeDutyCycle(7.5)

    time.sleep(1)

    pwm.ChangeDutyCycle(0)
```

This function rotates the motors duty cycle, sleeps for six seconds and then rotates back.

7.1.2 Infrared Sensor

To continue my goal of an autonomous project I used an Infrared sensor (IR). It's function is to check if the bin is full. The IR sensor I used was from the grove library so initially I had planned to use its documentation to control it[9]. However as I wasn't using the grove pi electronics board which was required to use the grove pi library I decided to use the GPIO library which I was already using to control the motors. This way I could test the GPIO pins directly and ensured my code was reusable.

```
if GPIO.input(23):
  print("Bin is empty")
else:
  print("Bin is full")
```

This tested the GPIO pin 23 directly to see if it is high or low. If the pin is high the bin is empty and if the pin is low the bin is full. I used an if loop to achieve this. When I was first testing this I had a few problems trying to get a high reading from the sensor. The sensitivity on the sensor was quiet low and I only got a reading when movement was made relatively close to the sensor. This was resolved by adjusting the potentiometer on the back of the sensor so it was more sensitive. This image below comes from the grove documentation, showing its inputs and outputs.

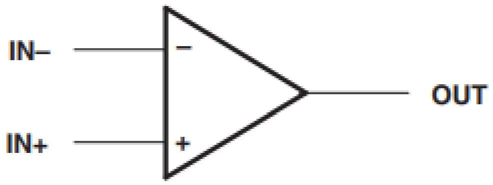


Figure 7.1.2-1

The IR above is from the grove documentation showing three pins, GND and VCC which is connected to the Raspberry Pi power pins and output which is connected to GPIO pin 23. For my specific IR sensor I also had a fourth pin N/C which I left unconnected.

7.1.3 DHT11 Sensor

A DHT11 sensor is used to detect temperature and humidity. I wanted to use this for my compost bin to ensure optimal composition is taking place. This way the user can monitor the compost on the webpage, once again ensuring the bin is self-operating. I created the following image on KiCad, showing GND, VCC, DATA pin and unconnected pin.

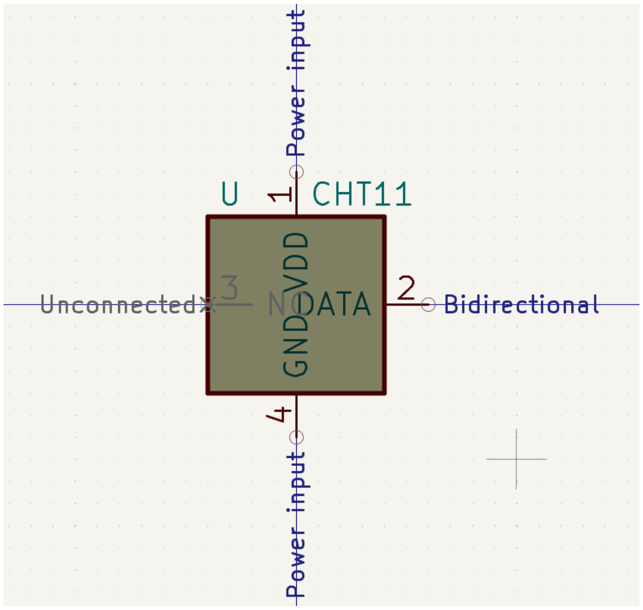


Figure 7.1.3-1

The sensor has 4 pin outputs, GND, which is connected to the GND of my Raspberry Pi. VCC is the power pin which is connected to the 5V Raspberry Pi pin[10]. SIG is connected to GPIO pin 24 which is where the signal is read and I have left NC untouched. Using the GPIO pins, I can read in the DHT11 sensor[11] by downloading the dht11 library. Temperature and humidity is then captured by the result. The following piece of code shows how I did this.

```
sensor = dht11.DHT11(pin=DHT_PIN)

result = sensor.read()

temperature = result.temperature
```

```
humidity = result.humidity
```

I didn’t want the temperature and humidity to be captured all the time so I added a delay. Now this function is read every 60 seconds saving power and avoiding a sensor overload. The following code captures the current time and when the dht11 is read, checks if it has been a minute, and then calls its function.

```
if time.time() - last_dht_read_time >= 60:
        temp_humidity()
       last_dht_read_time = time.time()
```

When I first tested this sensor on its own it was working and I was getting readings every minute. However when I added this functionality with the rest of my code I stopped getting readings. I decided to talk to my lecture about this who had good experience working with this sensor and I was told that there were problems reported when the sensor was pared with other functions. The sensor has a single wire protocol and relies on its own timing for the sensors data transmission. If another function is running, it can interfere with the DHT11 sensors and data will not be processed properly. This was the case for my DHT11 sensor and unfortunately I did not have time to get a new sensor. I decided to leave the sensor in my code but it does not give back any readings.

7.1.4 Raspberry Pi

The Raspberry Pi 4B is the microcontroller I have used to host my computer vision project. I opted for the 8GB RAM which is best for running heavy applications and multitasking which was appropriate for my AI applications compared to the 4GB RAM option. I decided to use a 32GB micro SD card to ensure I had enough space for the OS and for any applications I planned on using. The Raspberry Pi has a 40 GPIO pins which includes power pins, output pins, I^{2}C pins, SPI pins, UART pins and other functions. I focused on using the power pins and output pins for each of my inputs, my two sensors and my output components, the two servo motors. Using the GPIO pins directly like this reduces the need for additional microcontrollers or boards allowing real time communication between the components and the Raspberry Pi. Interaction between the components allows the Raspberry Pi to work as a self-contained system. The image below shows the output of the pins which I have used from KiCad.

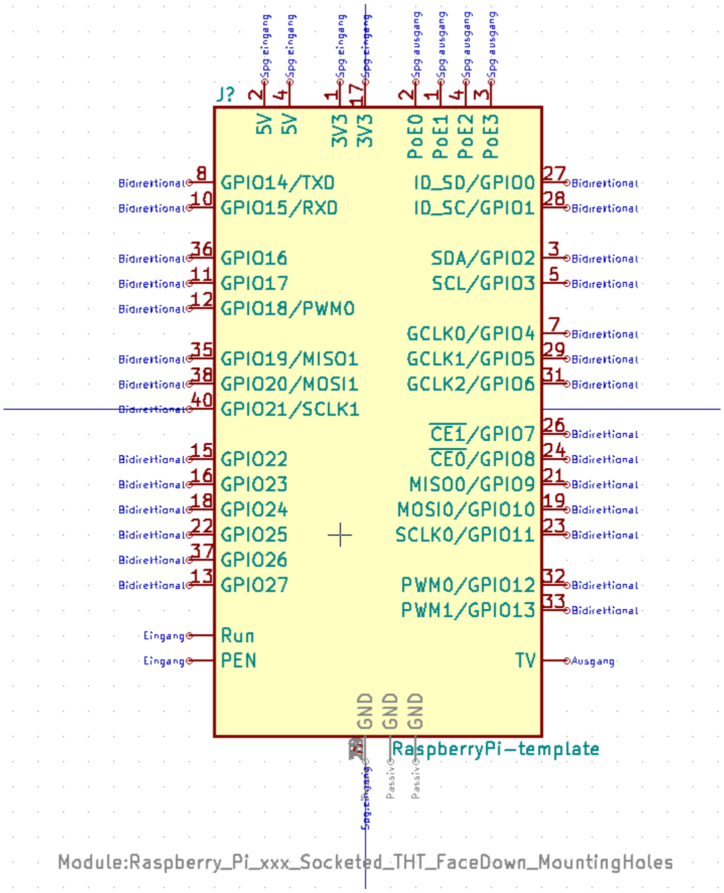

J?
2 5V Spg eingang
4 5V Spg eingang
1 3V3 Spg eingang
17 3V3 Spg eingang
2 PoE0 Spg ausgang
1 PoE1 Spg ausgang
4 PoE2 Spg ausgang
3 PoE3 Spg ausgang
Bidirectional 8 GPIO14/TXD
Bidirectional 10 GPIO15/RXD
Bidirectional 36 GPIO16
Bidirectional 11 GPIO17
Bidirectional 12 GPIO18/PWM0
Bidirectional 35 GPIO19/MISO1
Bidirectional 38 GPIO20/MOSI1
Bidirectional 40 GPIO21/SCLK1
Bidirectional 15 GPIO22
Bidirectional 16 GPIO23
Bidirectional 18 GPIO24
Bidirectional 22 GPIO25
Bidirectional 37 GPIO26
Bidirectional 13 GPIO27
Eingang Run
Eingang PEN
ID_SD/GPIO0 27 Bidirectional
ID_SC/GPIO1 28 Bidirectional
SDA/GPIO2 3 Bidirectional
SCL/GPIO3 5 Bidirectional
GCLK0/GPIO4 7 Bidirectional
GCLK1/GPIO5 29 Bidirectional
GCLK2/GPIO6 31 Bidirectional
CE1/GPIO7 26 Bidirectional
CE0/GPIO8 24 Bidirectional
MISO0/GPIO9 21 Bidirectional
MOSI0/GPIO10 19 Bidirectional
SCLK0/GPIO11 23 Bidirectional
PWM0/GPIO12 32 Bidirectional
PWM1/GPIO13 33 Bidirectional
TV Ausgang
GND Spg eingang
GND Passive
GND Passive
RaspberryPi-template
Module:Raspberry_Pi_xxx_Socketed_THT_FaceDown_MountingHoles

Figure 7.1.4-1

7.1.5 Camera

Initially, my plan was the use the Raspberry Pi camera to capture real time data. In order to use the Raspberry Pi camera alongside my Raspberry Pi I connected the flex cable to the Raspberry Pi[12]. My Pi's camera came pre-installed in the Raspberry Pi so the camera was ready to use. Unfortunately for my demo I will not be using my Raspberry Pi camera as it did end up getting damaged. I am using a regular USB camera instead which is connected directly to the Raspberry Pi USB port.

7.2 Software and Applications

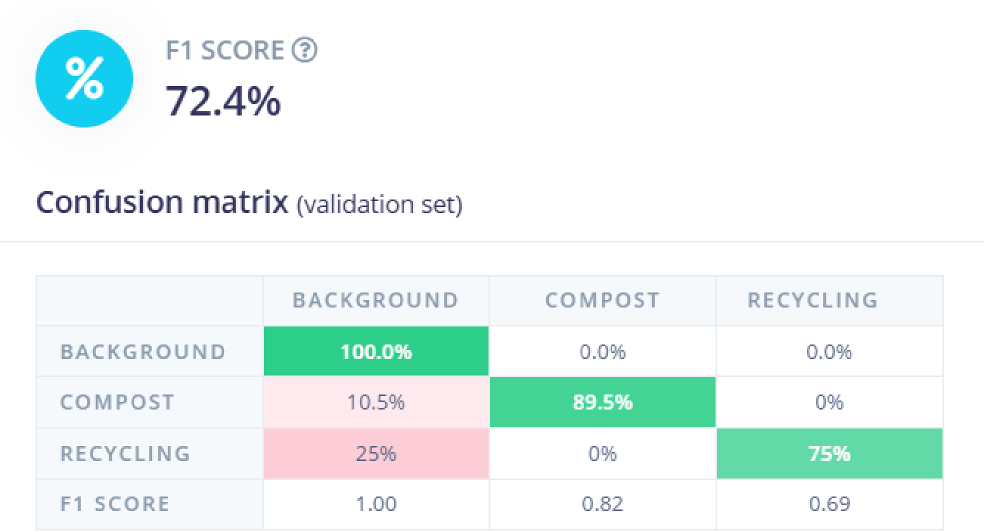
For this subsection I will go into greater detail exploring the software platforms and tools I have used in creation of my project. In each section I will talk about why I chose these applications and any technical difficulties experienced in the process.

7.2.1 Edge Impulse

Edge Impulse is an AI platform for building, testing and deploying machine learning models. This is what I used to train my waste management project. I uploaded my dataset to Edge Impulse and from there I was able to set the classifications, recycling or compost. I use object detection labelling which draws a box around the object I want to train. This means I can train the model to locate the object which prevents the model from picking up background images. I was able to sort my data into training and test data which I tried to hit a 80 to 20 ratio. With 541 data images processed, the next step was to create my impulse. This step was all about how I would turn my data into my machine learning model which for my case I used it for object detection. When this was created it showed me my two output classes. I decided to set the colour to RGB images which meant that my model would be able to process the colours when training the data. Now I could train my model and see the performance of it.

I worked on the neural network settings and these played a big role in the outcome of my score. As my dataset was quiet small, it was recommended to keep the number of training cycles between 20 and 100[13]. The number of epochs refer to the number of times the model learns from the dataset. If the number of epochs is too low it can reduce the accuracy as the model is not trained enough. However if the number is too high, it can cause oversampling which will also decrease the accuracy of the model. After much trial and error my model ran best at 60 epochs.

The learning rate controls how much the models parameters are updated during each step of the training process. In order to find the best rate I tested it at different rates starting with the standard rate of 0.001 and gradually increasing it. The higher the values the faster the learning rate. My model was more stable using a 0.01 rate. Overall, I was happy with my recycling and compost, receiving a 70% and 82% result for each of the classes.



F1 SCORE

72.4%

Confusion matrix (validation set)

	BACKGROUND	COMPOST	RECYCLING
BACKGROUND	100.0%	0.0%	0.0%
COMPOST	10.5%	89.5%	0%
RECYCLING	25%	0%	75%
F1 SCORE	1.00	0.82	0.69

Figure 7.2.1-1 Edge Impulse Model

Edges detection uses FOMO[14], Faster Objects More Objects, which is a machine learning algorithm which detects the location and the class of the objects. This means that if the object is large it can be picked up more than one time as it will detect multiple grids. FOMO worked well with my model and uses up to 30 times less processing power than other object detection devices. My model also uses MobileNetV2 0.35 which is a lightweight type of CNN, common neural network, used for feature extraction alongside FOMO.

In order to run my model, I use `edge-impulse-linux-runner` which deploys my model onto the Pi. With my camera connected I am able to see live data and can see what is being detected. Using the subprocess library, I am able to call this edge impulse cli command from inside my code, using standard output so the output can be read. I have used subprocess.Popen() which is a low level interface[15].

```
process = subprocess.Popen(
    ['edge-impulse-linux-runner'],
    stdout=subprocess.PIPE,
    stderr=subprocess.PIPE,
    text=True
)
```

The data from edge is sent in a JSON array. This is the output I was getting when a detection was being made.

```
[{"height":8,"label":"Recycling","value":0.640317440032959,"width":8,"x":48,"y":56}]
```

I needed to read the data inside of the square brackets and extract the data from label. In order to read the output I have used the process standard output library and printed the output[16].

```
line = process.stdout.readline()

print(line.strip())
```

In order to access which class is detected, I needed to extract the JSON, so I added a regex to do this by importing the library ‘re’[17]. This regex reads any JSON characters zero or more times inside a square bracket. I found working with regex evaluators helpful. If any JSON is found, I convert the JSON string into a Python object, by importing ‘json’ and using ‘json.loads’[18]. By using ‘match.group(1)’ it is matched to the output at ‘search’.

```
match = re.search(r'(\[.*\])', line)

 if match:

    try:

       detections = json.loads(match.group(1))
```

Compost and recycling are both saved under “label”. I have created a for loop which iterates through detections and gets the “label” value. If compost or recycling is detected it is printed to the screen.

```
             for obj in detections:

               label = obj.get("label")


                 if label in ["Recycling", "Compost"]:

                      if label == "Recycling":

                         print("Recycling detected, opening bin")
```

```
elif label == "Compost":
    print("Compost detected, opening bin")
```

Using FOMO means that an object may be detected multiple times which meant that the motor would be rotating continuously. To prevent this from happening I have added a delay after the first detection. The model picks up the first class detected, opens the bin and delays the motor for 10 seconds. In the meantime, detections are still being made but the motor will not move. Below is the if loop called before the detection.

```
if read_motor_time - last_movement_motor >= delay_for_motor:
```

7.2.2 Visual Studio Code

Initially the plan was to write my code using Pycharm, as this IDE is specifically built for Python which makes it easy to use for built in libraires. However as I was working on the Raspberry Pi it was extremely important that I chose an IDE which was lightweight. I was limited to CPU and RAM on the Raspberry Pi so I decided to use a different IDE as Pycharm was slow running on the Pi. Therefore I decided to use Visual Studio Code as my IDE, a lightweight and responsive platform for running my code.

7.2.3 Tiny DB Database

In order to store my data for my webpage, I needed a database. There were many different databases to choose from but I wanted to use a lightweight database for my Raspberry Pi. When running my computer vision application from Edge Impulse, the output was stored in JSON format. Therefore I wanted a database which could handle my JSON data which is why TinyDb was a perfect choice[19]. TinyDb is a lightweight NoSQL database. I imported TinyDb and created a class to handle the functionality of saving and retrieving the data. TinyDb automatically creates a place to store the data in db.json. Here I have defined a path where the data will be stored. In order to create an instance of this class and use it in other files I have used init and self.

```
def __init__(self, db_path="/home/ciara/Documents/FinalYrPro/db.json")
```

Here is the function I have created to save the data.

```
def detection(self, data):
    self.db.insert(data)
```

I imported this function into my main file and saved the data under the following headings, type, action and time. This is an example of how I saved data for when recycling was detected.

```
db.insert({
          "type": "Recycling",
          "action": "Opened bin",

          "time": time.time()

             })
```

The data is then saved in the db.json file. In the next heading I will talk about how I have accessed that data.

7.2.4 Flask

In order to make the data accessible to users, I wanted to make a frontend webpage to display my data. I decided to use Flask[20], a lightweight Python web framework to handle requests between TinyDB and my webpage. In order to start the application I imported the Flask module and initialised the database within flask. Upon starting up the project, I wanted the webpage to begin in a home page and from there I could route to different pages. I used the `@app.route('/')` to define the route. The backslash defined my homepage. I created another page for retreiving the data from TinyDB.

```
@app.route('/data')

 def index():

     data = db.get_detections()

  return render_template('index.html',
  data=data)
```

I called the function from TinyDB and returned render template so that the data would be returned in its HTML file.

I wanted to call my webpage inside of my main page when I was starting up my project. This is when I came across Threading. Threading is a model which enables a program to do more than one thing at the same time[21]. One part of my code will run Flask while the other part will handle my computer vision code. To do this I imported the library "threading" and created a new thread. Inside this I can tell it to run a new function and start the thread.

```
flash_thread = threading.Thread(target=run_flask)

flash_thread.start()
```

The function which I have called, "run_flask" is created inside my flask file. Inside this function I call my Flask application.

7.2.5 HTML and CSS

While Flask is used to handle the user data I needed to structure and style the webpage. When a function is called inside of the flask file, it is rendered to a HTML file. I have two HTML files and two CSS files. First is my home page which is where the user will be directed to once the project is up and running. The home page contains no data from the project just HTML and CSS. My second page contains the data from the project. In order to inject data from Flask into my HTML page, I have used Jinja2 template[22]. My data is stored inside of data which I have looped through these readings and displayed them in a table.

```
{% for item in data %}
```

```
<tr>
<td>{{ loop.index }}</td>
<td>{{ item['type'] }}</td>
<td>{{ item['action'] }}</td>
<td>{{ item['time']  | datetimeformat}}</td>
</tr>
{% endfor %}
```

Using this loop I can iterate through the each record and display the data. I have stored this data inside a table and so the table is updated when a detection is made. To page is refreshed every 5 seconds so the user doesn’t have to continually refresh the page when data is updated. To style this webpage, I have used CSS[23]. I found W3Schools useful for testing how I wanted my webpage to look.

8 Ethics

Why do we need to consider ethics when developing a project? Ethics is important because it ensures peoples safety, fairness and equality[24]. It helped me to consider the real world impacts of my project and how it could affect other people. From the start ethics was a huge consideration of mine as I wanted to create a project which would positively impact society.

Throughout my report I talk a lot about how I wanted to use low powered components. I wanted to use an energy efficient and affordable microcontroller. The Raspberry Pi was a suitable microcontroller for hosting my machine vision project which required extensive features. Instead of hosting it to the cloud it is deployed locally from Edge Impulse to the Raspberry Pi which uses less energy over time.

It was important to me that my components used as little power possible when running. Using a servo motor was far more efficient than using a DC motor. Power is only being used when the servo motor is rotating. The combination of the temperature and humidity sensor into one component means that I save space and power. The DHT11 sensor uses very little power and polling the sensor every minute conserves the power furthermore.

Edge Impulse has a number of ethical practises when working on their platform. My project is based on a sustainable system designed to promote responsible behaviour by reducing harm to our environment. Their platform has free access to users making it accessible to everyone. By using their lightweight and power efficient models, FOMO and MobileNetV2 0.35 I minimise my carbon footprint.

I gave thought to the code I was using. TinyDB’s low memory usage and minimal processing power made it a great choice for my project when requiring small amounts of data. I wanted to use a minimal interface which would allow me to interact with the database and webserver. One of the main reasons for choosing Flask is because it is a micro web frame, making it ideal for my application.

Overall, my project promotes a wide range of ethical practises with its main focus on supporting the environment.

9 Conclusion

I have created an autonomous computer vision waste detection project which sorts recycling and compost into the correct bin. Using features to reduce the need for manual input, I have added two servo motors to open and close the detected bin while the use of an IR sensor tells the user if the bin is full. The data is stored in a database, managed by Flask and displayed using CSS and HTML. I have successfully created a sustainable environmentally friendly independent project using both my hardware and software components.

The central theme of my project was computer vision. Without this, objects wouldn’t be classified into the right bin. Overall, I was happy with the score of my model stated in my report above. Everything I tested was picked up by the model and classified into the correct class. However if I was given more time I would have liked to build a bigger class. I do believe my score would have increased. That said using a small dataset was ideal for my project.

10 References

[1] Fei-Fei Li, “Artificial intelligence is not a substitute for human intelligence; it is a tool to amplify human creativity and ingenuity” quoted in [Online]. Available: https://unu.edu/article/we-need-effective-governance-shape-ai-good#:~:text=“Artificial%20intelligence%20is%20not%20a,the%20pursuit%20of%20AI%20itself.

[2] Alex Calder, “Beyond the Bin: We need to rethink our relationship with waste – M-CO.” Mco.ie. Accessed: Apr. 02, 2023. [Online]. https://mco.ie/beyond-the-bin-we-need-to-rethink-our-relationship-with-waste/

[3] U.S. Environmental Protection Agency, “Recycling Economic Information (REI) Report” EPA, [Online]. Available: https://www.epa.gov/smm/recycling-economic-information-rei-report#:~:text=Recycling%20also%20conserves%20resources%20and,to%20collect%20new%20raw%20materials.

[4] P. Proença, TACO*:* Trash Annotations in Context Dataset, GitHub repository, 2020. Accessed: November 2024 [Online]. Available: https://github.com/pedropro/TACO

[5] S. Frost, CompostNet, GitHub repository, 2021. Accessed: January 2025 [Online]. Available: https://github.com/sarahmfrost/compostnet

[6] Yaniv Noema, “Top 3 Programming languages For Implementing a Computer Vision System. ” Medium. Accessed: October 2024 [Online].https://medium.com/imagescv/my-top-3-programming-languages-for-implementing-a-computer-vision-system-2aa0a3ad0a2a

[7]Raspberry Pi, “Getting started with your Raspberry Pi”. Accessed: November 2024 [Online].https://www.raspberrypi.com/documentation/computers/getting-started.html

[8] Grandpa Scarer, “Using a servo.” Raspberrypi.org. Accessed: 2024. [Online]. https://projects.raspberrypi.org/en/projects/grandpa-scarer/3

[9] Seeed Studio, “Grove – Infrared Reflective Sensor”. Accessed: 2025. [Online]. https://wiki.seeedstudio.com/Grove-Infrared_Reflective_Sensor/

[10] Seeed Studio, “Grove – Temperature&Humidity Sensor (DHT11)”. Accessed: 2025. [Online]. Grove - Temperature&Humidity Sensor (DHT11) | Seeed Studio Wiki

[11] Pypi, “DHT11 Python library”. Accessed: 2025. [Online]. dht11 · PyPI

[12] Raspberry Pi, “About the camera modules”. Accessed: 2025. [Online]. https://www.raspberrypi.com/documentation/accessories/camera.html

[13] Edge Impulse, “Learning Blocks”. Accessed: 2024. [Online]. https://docs.edgeimpulse.com/docs/edge-impulse-studio/learning-blocks

[14] Edge Impulse, “FOMO: Object detection for constraint devices”. Accessed: 2024. [Online]. https://docs.edgeimpulse.com/docs/edge-impulse-studio/learning-blocks/object-detection/fomo-object-detection-for-constrained-devices

[15] Moez All, “An introduction to Python Subprocess: Basics and Example”, DataCamp. Accessed: 2025. [Online]. https://www.datacamp.com/tutorial/python-subprocess

[16] deft_code, “read subprocess stdout line by line”, StackOverflow. Accessed: 2024. [Online]. https://stackoverflow.com/questions/2804543/read-subprocess-stdout-line-by-line

[17] Google, “Python Regular Expressions”. Accessed: 2025. [Online]. https://developers.google.com/edu/python/regular-expressions

[18] W3Schools, “Python JSON”. Accessed: 2025. [Online]. https://www.w3schools.com/python/python_json.asp#:~:text=Parse%20JSON%20-%20Convert%20from%20JSON,will%20be%20a%20Python%20dictionary.

[19] Python Engineer, “TinyDB in Python – Simple Database For Personal Projects”. Accessed: 2025. [Online]. https://www.python-engineer.com/posts/tinydb/

[20] GeeksForGeeks, “Flask Tutorial”. Accessed: 2025. [Online]. https://www.geeksforgeeks.org/flask-tutorial/

[21] Site24x7, “Introduction to Threading in Python”, ManageEngine. Accessed: 2025. [Online]. https://www.site24x7.com/learn/threading-in-python.html

[22] GeeksForGeeks, “Flask Rendering Templates”. Accessed: 2025. [Online]. https://www.geeksforgeeks.org/flask-rendering-templates/

[23] GeeksForGeeks, “How to use CSS in Python Flask”. Accessed: 2025. [Online]. https://www.geeksforgeeks.org/how-to-use-css-in-python-flask/

[24] Alexandra Dauchess, “Understanding the importance of Ethics in Information Technology”, MaryMount.edu. Accessed: 2025. [Online]. https://marymount.edu/blog/understanding-the-importance-of-ethics-in-information-technology/#:~:text=Key%20Ethical%20Principles%20in%20IT&text=Honesty%20—%20Information%20technology%20professionals%20are,and%20will%20have%20on%20society