Smart Cycle

FYP Project Demo

Ciara Crowe

Supervisor: Niall O'Keefe

Introduction

Waste detection project which uses computer vision to classify objects into waste and compost.

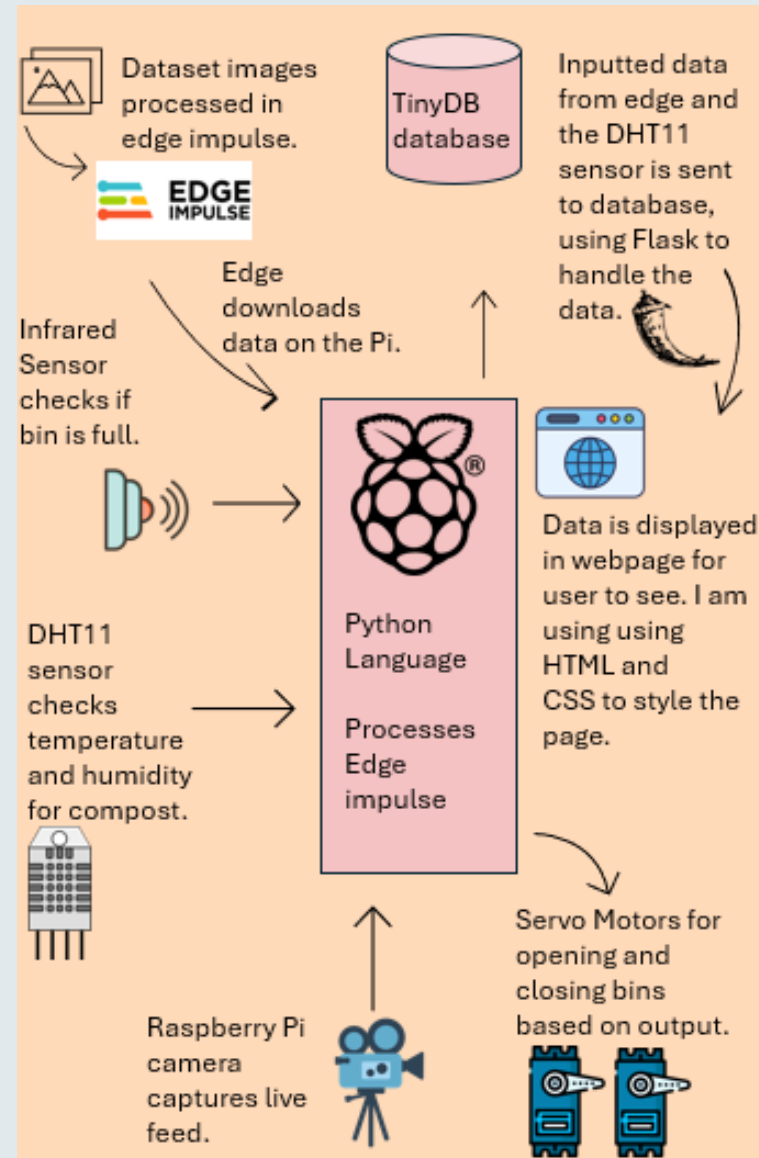Edge Impulse – Uploaded data, applied image classification and object detection. Deployed locally to Raspberry Pi.

Servo Motors, Infrared Sensor and DHT11 Sensor. Data saved in TinyDB.

Data is displayed on webpage. Flask, HTML and CSS

Architectural Diagram

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.

Infrared Sensor checks if bin is full.

Edge downloads data on the Pi.

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.

Raspberry Pi camera captures live feed.

Servo Motors for opening and closing bins based on output.
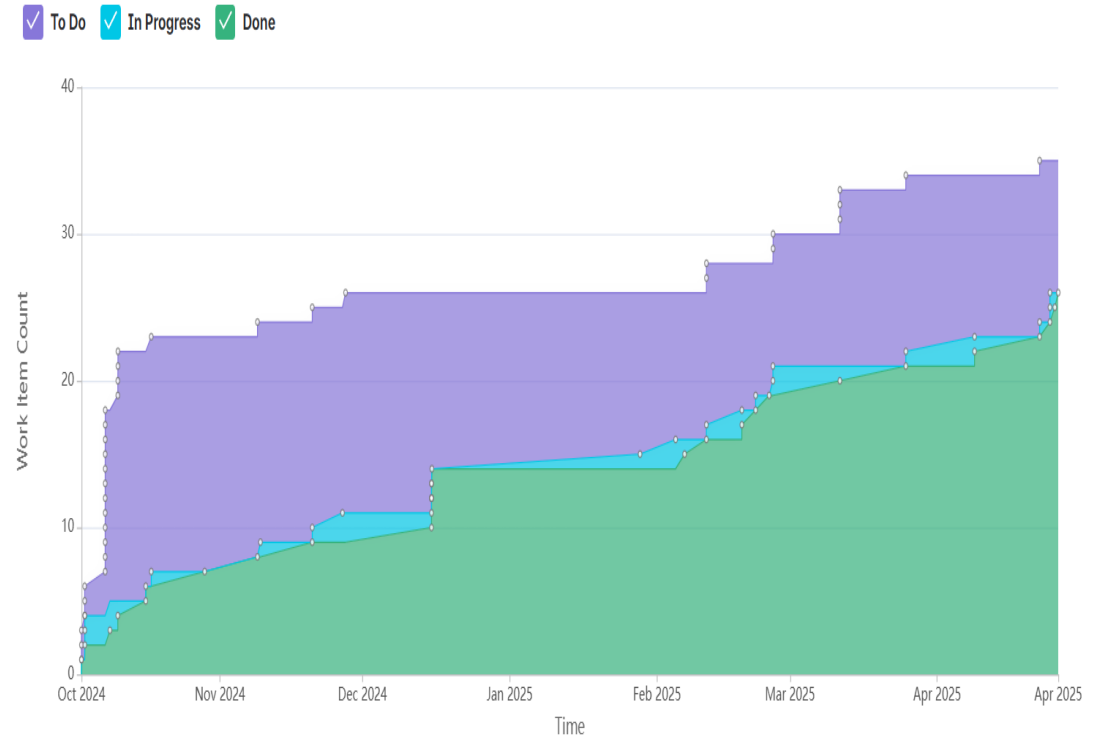
Project Management JIRA

Sprints

Failures

Epics

Backlog

Research

- Raspberry Pi
- Setting up OS – sd card/pi imager
- Connections to other peripherals
- Computer Vision
- Edge Impulse
- Dataset
- Object Detection (bounding boxes)
- Model evaluation

Technical Content

- Python, HTML and CSS

- Raspberry Pi

- Edge Impulse – computer vison platform

- Servo Motors

- Infrared Sensor

- DHT11 Sensor

- TinyDB

- Flask

Edge Impulse

- Locally deployed my edge impulse model to my raspberry pi – Edge impulse CLI impulse runner.

- Used CompostNet dataset

- Two classes waste and recycling

- Uses FOMO and MobileNetV2 0.35

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

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

```python
while True:
    line = process.stdout.readline()

    print(line.strip())

    match = re.search(r'(\[.*\])', line)
    if match:
        try:
            detections = json.loads(match.group(1))

            read_motor_time = time.time()

            for obj in detections:
                label = obj.get("label")
```

Servo Motors and IR Sensor

- When a detection is made a servo motor rotates depending on what class is detected.
- If the bin is full a high is detected which stops the motors from opening and closing.

```python
RECYCLING_PIN = 24
COMPOST_PIN = 16
```

```python
GPIO.setup(RECYCLING_PIN, GPIO.OUT)
GPIO.setup(COMPOST_PIN, GPIO.OUT)
```

```python
pwm_recycling = GPIO.PWM(RECYCLING_PIN, 50)
pwm_compost = GPIO.PWM(COMPOST_PIN, 50)
```

```python
def open_bin(pwm):
    if is_bin_full():
        print('Bin is full')
        db.insert({
            "type": "Detection",
            "action": "Bin is full",
            "time": time.time()
        })
        return
    time.sleep(0.1)
    print("Opening bin")
    pwm.ChangeDutyCycle(5.0)
    time.sleep(3)
    print("Holding bin")

    print("Closing bin")
    pwm.ChangeDutyCycle(7.5)
    time.sleep(3)
    pwm.ChangeDutyCycle(0)
```

DHT11 Sensor

- Temperature and Humidity readings
- Runs every minute

- Not reading properly when connected alongside other functions.

```python
DHT_PIN = 26

sensor = dht11.DHT11(pin=DHT_PIN)

def temp_humidity():
    result = sensor.read()
    temperature = result.temperature
    humidity = result.humidity
    print(f"Temperature: {temperature}°C, Humidity: {humidity}%")

    db.insert({
        "type": "Temperature and Humidity",
        "temperature": temperature,
        "humidity": humidity,
        "action": f"Temperature: {temperature}°C, Humidity: {humidity}%",
        "time": time.time()
    })
```

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

Dataset

- TinyDB – NoSQL
- Lightweight
- JSON

```python
class ProjectBinDatabase:
    def __init__(self, db_path="/home/ciara/Documents/FinalYrPro/db.json"):
        self.db = TinyDB(db_path)

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

    def get_detections(self):
        return self.db.all()
```

Flask

- Lightweight python web framework to handle requests between database and webpage.

- Used routing to route to different pages

- Called my functions defined in database to retrieve data.

```python
app = Flask(__name__)
db = ProjectBinDatabase("/home/ciara/Documents/FinalYrPro/db.json")


@app.route('/')
def home():
    return render_template('home.html')


@app.route('/data')
def data():
    data = db.get_detections()
    return render_template('data.html', data=data)


@app.route('/live')
def live():
    return redirect("http://192.168.178.155:4912/")
```

Threading

- I wanted flask to run with my backend

- Threading – runs multiple processes at the same time

- Created a new thread and called the function.

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

HTML & CSS

- Structure and styling
- Jinja2

```
{% for item in data %}
<tr>
    <td>{{ loop.index }}</td>
    <td>{{ item['type'] }}</td>
    <td>{{ item['action'] }}</td>
    <td>{{ item['time'] | datetimeformat}}</td>
</tr>
{% endfor %}
```
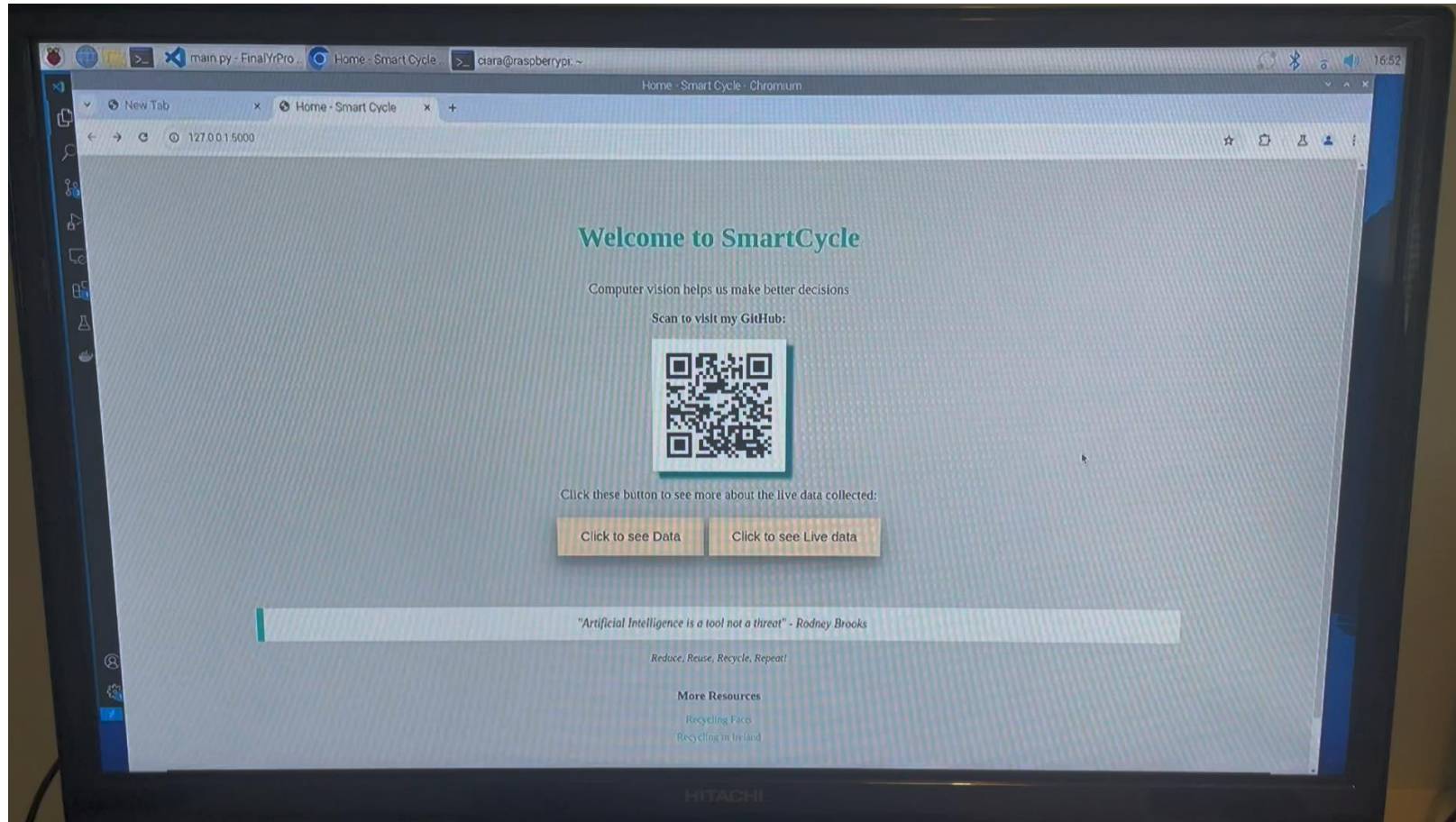
```css
body {
    font-family: 'Papyrus', fantasy;
    background-color: #D3D3D3;
    text-align: center;
    padding: 50px;
}


h1 {
    color: #20B2AA;
    margin-bottom: 40px;
}


table {
    width: 80%;
    margin: auto;
    border-spacing: 0;
    box-shadow: 10px 10px 5px rgb(0, 128, 128);
    background: white;
    border-radius: 8px;
}
```

Video

Future Improvements

Add more data

Add more classes

Check out my Github

- https://github.com/CiaraC03/FinalYrProject