An IOT connected Solid Waste Sorting System

A Project Proposal by Niklas Harnish

Abstract

In today's world of rapid consumption, plastic waste has become more of a problem than ever. That is why this project is setting out to build an open-source IoT system, with a plastic classification module. This IoT system will be a platform for other developers to build on, and will be able to collect and display data from the plastic classification module.

1. Introduction

This project proposal is split into six parts. The background section contains an analysis of software/projects that aim to do similar things to this project. Their strengths and weaknesses are then listed. The motivation then contains information on gaps within current systems that can be improved upon, or that don't fulfill the needs of a certain market. Within the scope section the aims and goals of this project are described, and perhaps more importantly, what this project **does not** want to be is also described. These are then expanded into formal functional and non-functional requirements, within the next section, requirements. Next comes methodology, where different technologies that are to be used, and the method of feedback collection is described. Finally, a formal timeline with some analysis on important steps throughout the project is given, along with a Gantt chart for clarity.

2. Background

With global plastic production at an all-time high (390 million metric tons in 2021 [13]), plastic sorting has become more and more important. At the same time, the average customer can not distinguish between different types of plastic and even if they could, the likelihood of customers being willing to have a different bin for each color, or type of plastic is low. As the world grapples with the consequences of the plastic waste generated through our ever increasing consumption, the need for a smart and connected system that requires little user intervention increases. However, plastic classification software and hardware often comes at great (monetary) cost.

The AUTOSORT FLAKE (AS Flake), created by the Norwegian for-profit company Tomra, is a high-performance flake sorting system. It can sort *"all kinds of polymers, including silver and white, as well as wood and paper"* and for metals *"all kinds including non-ferrous*

metals" [5]. With a plastic sensor resolution of 2 mm and metal sensor of 1 mm it can detect very small flakes. However, due to the proprietary nature of this product, there is not much publicly available information on both the capabilities or the price of the AS Flake, however it can be assumed by the size and industrial applications of this device that it is quite expensive. This will hinder many potential customers from buying it, it is only really useful within an industrial setting.

In contrast the *Plastic Scanner* is an open source plastic classification project, with instructions on how to create the hardware and to build and install the firmware hosted on GitHub[8]. The project is currently still in development, with the website stating that it is not ready to be used and should not be built unless seeking to help develop the product [9]. However, once in a production ready state, they promise to be able to identify five different types of plastics using a small handheld device. Internally, the device uses *"discrete near-infrared (NIR) spectroscopy"* [10] to identify plastics. Additionally the website has precise instructions on how to contribute to the project. While the Plastic Scanner is a great project, as mentioned it is still in development and also has a more niche application, mainly being used for handheld plastic identification. It is missing some automation features that might make it more generally useful.

OpenSpecy, described by Analytical Chemistry as open-source software that *"allows users to view, process, identify, and share their spectra to a community library"*. [11] These processed spectra can then be used with other programs. As described above with the *Plastic Scanner*, spectroscopy is often used to identify plastics, making this a valuable solution for this step in plastic classification. While the program can be used online, it also has a library which can be used with the popular statistical computing language "R", to process spectra programmatically. However, it is not clear if these spectra are processed locally, on the user's computer or within a server hosted by OpenSpecy. OpenSpecy has an open-source repository, hosted on GitHub, which describes the process of installation and using the program. [12] Theoretically any developer could use open-specy to create whatever plastic classification system they want. However, it is simply a small (but very important) part of plastic classification and a system has to be built around it. This requires a great deal of knowledge and expertise, in both spectroscopy and in software development.

The *Home Assistant* is an open-source smart home system that focuses on "Local Control and Privacy" [1]. It lets the user add their existing smart home products, from reputable brands such as *Phillips Hue* and *Sonos* [2], to a single interface. With 2569 integrations [2] many users will be covered by their supported products. Additionally, there is extensive documentation on how to install the software on the user's devices, with a range of supported options, such as on a Raspberry Pi, or within a Docker Container [3]. All the code for this project is available on a public *GitHub* account [4], with clear guidelines for contributing listed in the documentation [3]. This issue with this project is that it is meant to integrate with existing technologies, with developing for the platform being a secondary objective.

The proposed project will aim to create two separate, but connected systems. The first is an open-source platform for building IoT modules. This will also include a hub, to connect these

modules and an easy to use frontend for easy control by the end user. Along with this, documentation will be provided, so developers can replicate and expand the module ecosystem, or develop on top of the foundations built by this project. The second is an open-source, IoT connected plastic classification device, that can be used to collect data and distinguish between different plastics.

3. Problem statement

As briefly mentioned above, the existing solutions have some issues that leave a gap in the market. While Home Assistant is a good project for people interested in connecting their existing smart home devices, it is not meant as a platform to build on top of. While the AUTOSORT FLAKE provides a good experience to those that can afford, and have space for the device, it is not a good solution for normal people interested in doing their part. The Plastic Scanner is a good solution, but is not ready for use and does not have some required automation/interconnectivity features. Finally OpenSpecy is simply a tool for plastic analysis, it does not provide much of the infrastructure to actually use it.

Cause	Effect
1. No open source IoT system for knowledgable people to build on **2. Current solutions meant to integrate with existing, proprietary devices** **3. Plastic classification/sorting devices have high associated cost and space requirements** **4. Commercial Plastic classification/sorting devices use a lot of space** **5. Open-source classification software is still in development, no available product** **6. Open spectrometry software is available, however not in a complete classification package**	1. Lack of interest in building open-source IoT technology and devices, too big barrier of entry 2. Lack of incentive in building an open source device, as open source projects will integrate with proprietary devices 3. Smaller companies, communities cannot afford to sort plastics, leaves plastic that could be sorted but isn't 4. Many smaller companies/communities do not have any space to put these devices 5. People who are interested in the space need to use unfinished solutions, or need to use proprietary, expensive solutions 6. Developers need to have a deep understanding of spectrometry to use the information available, to build a classification system

4. Scope

This project will...

Create an IOT platform, which:

- Should include...
    - A Frontend for the user to interact and control their IOT devices with
    - A Backend, with which the frontend will communicate. This will have an API for the IOT modules to communicate and register themselves with
    - An interface for saving data from modules in a user defined database
    - Documentation on how to set up the platform, how to create user defined modules and how to interact with the API endpoints
    - The code hosted on a Public Git Repository
    - Simple example modules, that display how to create IOT devices
    - Refined documentation that has been adjusted based on hobby user feedback
    - Be relatively cheap to reproduce for yourself
- Should not include...
    - A general IOT platform, that connects with devices made for other systems
    - A “ready out of the box” experience, instead it is meant for tinkerers and people who want to build their own modules

Build a module for the IOT platform that:

- Should include...
    - A way of determining and distinguishing between different plastics, these could include color or material
    - The code hosted on a Public Git Repository
    - Documentation on how this module was made, which can be used as reference by other developers
    - Documentation on how to build this module yourself
- Should not include...
    - A high-performance system, useable for industrial purposes
    - A library for general plastic analysis
    - A product for sale

5. Requirements

These are ranked by Priority

The IOT System

Functional Requirements:

1. **Shall** have a backend that manages connected modules
2. **Shall** have a way to connect to new modules on the network
3. **Shall** have an API with which connected modules can communicate with
4. **Shall** have a frontend, with which the user can interact, and control connected modules with
5. **Shall** have an interface with which to communicate with a user-defined database
6. **Should** have example modules written for it, showing basic functions of the system
7. **Should** notify users of errors or crashes throughout the system, on the frontend so the user can easily fix any issues
8. **Should** include a language library which programmers can use to easily create modules for the system

Non-Functional Requirements

1. **Shall** be an open-source project
2. **Shall** have documentation that contains information on how to set up the IOT hub
3. **Shall** have documentation that describes the process of writing a custom IOT module
4. **Shall** be able to run on a Raspberry Pi Model 4b, with multiple IOT devices connected at once
5. **Shall** be able to be installed on a machine comparable to the Raspberry Pi Model 4b, running Linux
6. **Shall** use end-to-end encryption to protect user information
7. **Shall** be easy to set up, and write modules for, by an experienced software developer
8. **Shall** run for extended periods of time without crashing, should be able to recover from some errors
9. **Should** be cheap for an end-user to reproduce for themselves
10. **Should** be able to be easily installed in a docker container

Plastic Sorting System

Functional Requirements

1. **Shall** be written as a module to the IOT system
2. **Shall** be able to distinguish between different plastics, using some sort of camera or sensor
3. **Shall** keep track of internal statistics, such as amount of each plastic sorted
4. **Shall** have a way of selecting the plastic to be tracked, through the IOT frontend
5. **Shall** save data to the database provided to the IOT system

Non-Functional Requirements

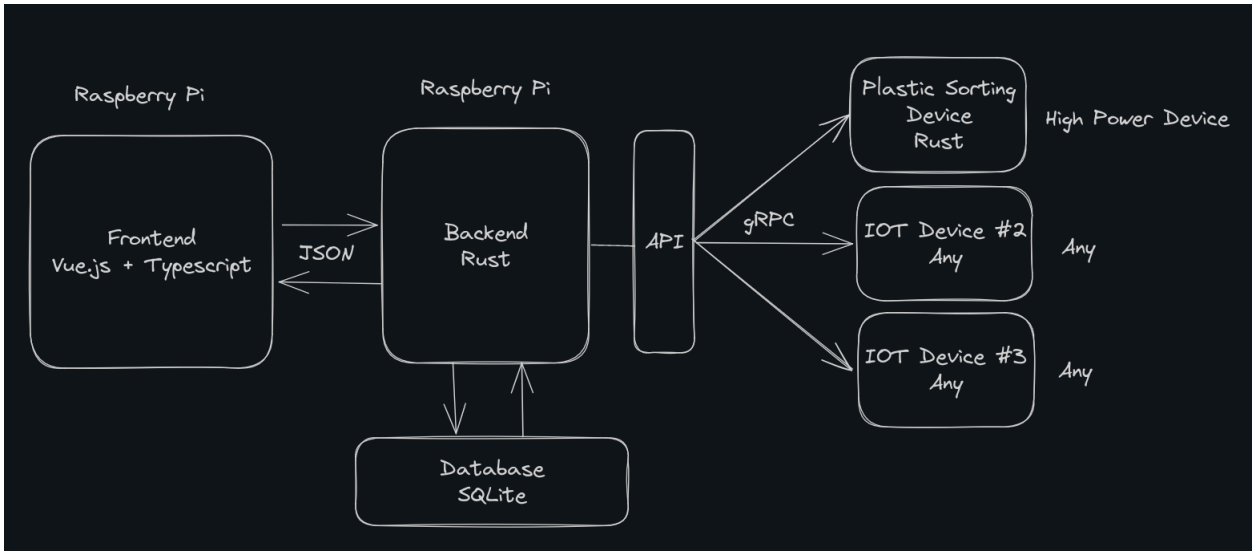
1. **Shall** have its code published on a public Git repository
2. **Shall** have extensive documentation, on how the user can build it for themselves, including a recommended parts list

3. **Shall** have documentation on how it was made, so it can be used as an example module by IOT platform developers

6. Methodology

As the main aim of this project is to build reliable, performant and scalable systems, choosing the right technology and development methodology is of particular importance. In **Figure 1**, a diagram of the architecture of the IOT System has been laid out. To complete this project each part of this diagram will need to be completed. This section will discuss some parts of this diagram, the technology that has been chosen to support them and the plan for completing them. It will also go over how the success of the totality of the project, but also individual parts, will be evaluated.

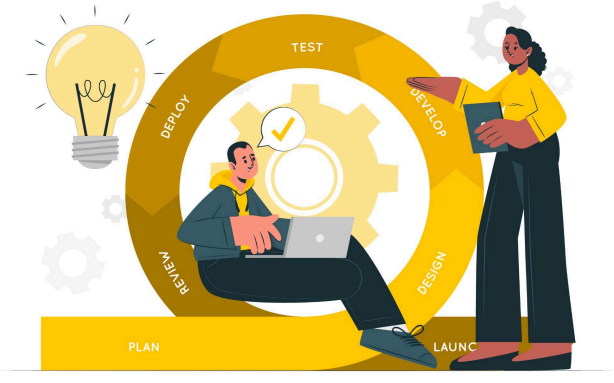
Figure 1:



Development Process

To ensure that this project is completed in an efficient and orderly way, agile development has been chosen. Agile fits this project well, as it is composed of multiple parts, which could fit into a sprint. Additionally, if during the development of the product requirements change, such as time constraints, agile development allows for faster requirement re-evaluation.

Figure 2: (image: Freepik.com)



Backend & IoT Devices

The main programming language for this project will be Rust, a safe, low-level and fast language. Rust has become popular in recent years due to its ability to be memory-safe (unless explicitly requested), without the overhead of garbage collection. This is quite important, due to the performance constraints introduced by IoT systems and these systems often being run for indefinite periods of time. The main competitor to Rust for this project would be C++, which certainly has its upsides. It is a mature language, with many resources and easy low-level memory access. However, due to Rust's memory-safety features and personal familiarity with the language it has been chosen for the backend for both the IOT and plastic distinguishing system.

Networking and Communication

To allow for easy communication between IoT modules and the backend-api *Remote Procedure Calls* (RPC) has been chosen, specifically gRPC. RPC allows for efficient and uncomplicated communication between internet devices, while abstracting away concepts such as sockets and data-interchange formats. GRPC is a popular, open-source, high performance RPC framework, with support for many languages.

Developing an Open Source application

To meet the requirement of *"shall be an open-source project"*, a number of established open-source conventions will be used. Two strong examples of such are the MIT and Apache licenses. The MIT license gives permission to use the provided code without restriction, allowing another developer to resell it or use it however they please [6]. In contrast, the apache license is more closed off, forcing derivative or 'recipients of this work' to also use the apache license, making that work also open-source [7]. In addition to choosing an open-source license, another important step is publishing the code in a public repository. For this a Github repository has been chosen, as it is one of the most popular and easy to use choices, while being free.

Testing

Non-functional requirements testing will be done in a multitude of ways. *"Shall be able to run on a Raspberry Pi Model 4b with multiple IOT devices connected at once."* This will be tested with performance testing. An example of such a test will include connecting many simulated clients, to test how many clients my backend could theoretically handle while running on the specified Raspberry Pi model. Another could include testing how many frontend clients sending requests could be handled at once. While this is not necessarily a good test for simulating a real life scenario, as normally the client will not have many frontend clients connected, it is a good measure of what is theoretically possible. Another performance test will include a mixture of both the mentioned tests, where some amount of frontend clients interact with the backend, while many IOT devices are also connected.

Another important aspect of the product is the user experience. To test if: *"shall be easy to set up, and write modules for, by an experienced software developer"* is fulfilled, a number of

peers will need to be used to collect feedback. One could test this through asking them to read through the documentation and testing them on their understanding. Another could be having a number of them try to set up the system themselves.

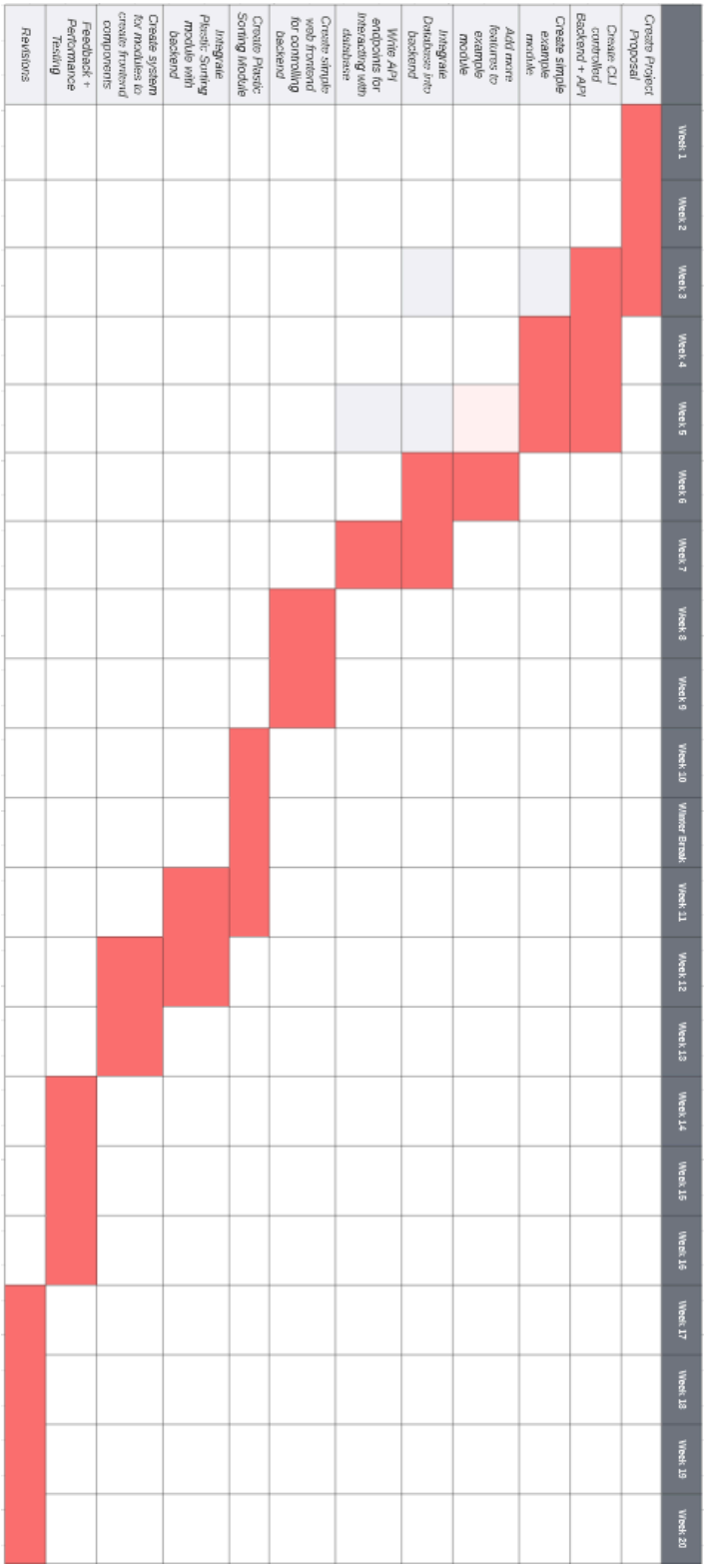
Functional testing will be done by evaluating if each functional requirement has been met. This will be done through asking peers to use the software and answer a questionnaire about the features that were available.

7. Timeline

To achieve the goals of this project, a well regimented work schedule will need to be followed. To create this schedule, each part of **Figure 1** has been split into appropriate steps. To view all these steps, see the Gantt chart in **Figure 2.** Below some steps of particular importance are highlighted. *Note that steps do not need to be a finalized state at the end of their allotted time, as there may need to be changes to their design as the project progresses*

- **Create CLI controlled Backend + API:** this contains creating the server, which allows IoT modules to connect to it. This server will run from a linux command line on the Raspberry Pi. It will also contain preliminary Api endpoints for requests from future IoT modules, however these will likely be changed in the future.
- **Create a simple example IoT module:** this module should be used to test and demonstrate the capabilities of the server and help to understand how API endpoints need to be changed. It will also help to get familiarized with the equipment.
- **Create a simple web frontend for controlling backend:** this part of the project will contain the process of making the example modules controllable through a simple GUI hosted on the web.
- **Create the Plastic Classification Module:** using what has been learned from the example modules, a plastic classification module will be created. This will take a significant amount of time and research, so it has been allocated over the winter break and given three extra weeks of time.
- **Create a system for modules to create frontend components:** an important part of the web frontend is flexibility. To enable that every IoT module will work with the platform a way for modules to specify their own web components needs to be created. The example modules will then be switched to this more dynamic method, instead of being hardcoded into the frontend.
- **Feedback + Performance Testing:** during the feedback stage, usability feedback on both the final product, but also the documentation will be collected. Additionally performance testing will be done
- **Revisions:** using feedback collected during the previous section, changes to the product and documentation will be made during this time.

Figure 3 Gantt chart:



8. References

[1] Home Assistant. “Home Assistant.” *Home Assistant*, 22 Oct. 2023, www.home-assistant.io/.

[2] Home Assistant. “Integrations.” *Home Assistant*, www.home-assistant.io/integrations/. Accessed 26 Oct. 2023.

[3] Home Assistant. “Installation.” Home Assistant, 2022, www.home-assistant.io/installation/.

[4] “Home Assistant.” GitHub, github.com/home-assistant. Accessed 21 Oct. 2023.

[5] “AUTOSORT FLAKE.” Hubspot, TOMRA, f.hubspotusercontent00.net/hubfs/4847902/AUTOSORT%20FLAKE%20Flyer_GB.pdf. Accessed 20 Oct. 2023.

[6] “The MIT License.” Open Source Initiative, 31 Oct. 2006, opensource.org/license/mit/.

[7] “Apache License, Version 2.0.” Open Source Initiative, 31 Oct. 2006, opensource.org/license/apache-2-0/.

[8] “Plastic-Scanner.” GitHub, github.com/Plastic-Scanner/. Accessed 26 Oct. 2023.

[9] Plastic Scanner. | Plastic Scanner. 10 Dec. 2020, plasticscanner.com/.

[10] “How It Works | Plastic Scanner Documentation.” Docs.plasticscanner.com, 2 June 2023, docs.plasticscanner.com/how_it_works. Accessed 26 Oct. 2023.

[11] Cowger, Win, et al. “Microplastic Spectral Classification Needs an Open Source Community: Open Specy to the Rescue!” Analytical Chemistry, vol. 93, no. 21, May 2021, pp. 7543–48, https://doi.org/10.1021/acs.analchem.1c00123. Accessed 28 Feb. 2022.

[12] PhD, Win Cowger. “Open Specy 1.0.” GitHub, 21 Oct. 2023, github.com/wincowgerDEV/OpenSpecy-package. Accessed 26 Oct. 2023.

[13] PlasticsEurope (PEMRG). "Annual Production of Plastics Worldwide from 1950 to 2021 (in Million Metric Tons)." Statista, Statista Inc., 2 Dec 2022, https://www.statista.com/statistics/282732/global-production-of-plastics-since-1950/