# What we wanted to do

With the threat of nuclear war becoming more and more probable, we took inspiration to create something which could help significantly in a nuclear war scenario, or any hazardous scenario which could happen.

We set out to achieve creating a Hazard detecting device, which could sense various environmental variables, such as: Heat, Light and Radioactivity. Since this would be detecting hazards, it would be important for the operator to stand outside of the room and still see what is happing, so we also would incorporate a camera vision system. All these technologies would then come together to server hosted on a laptop/pc with a GUI showing all the sensor data, with a live feed of the robot’s point of view.

# What we did

Overall, I think we achieved what we set out to do, this was achieved by combining a varying number of technologies that we had not yet used, ranging from using a Geiger counter to detect radioactive material, to creating a GUI in python, gathering various data points, and displaying them to the user and allowing a live feed from a mobile phone camera to the server/GUI. As you can see in figure one everything came together nicely.

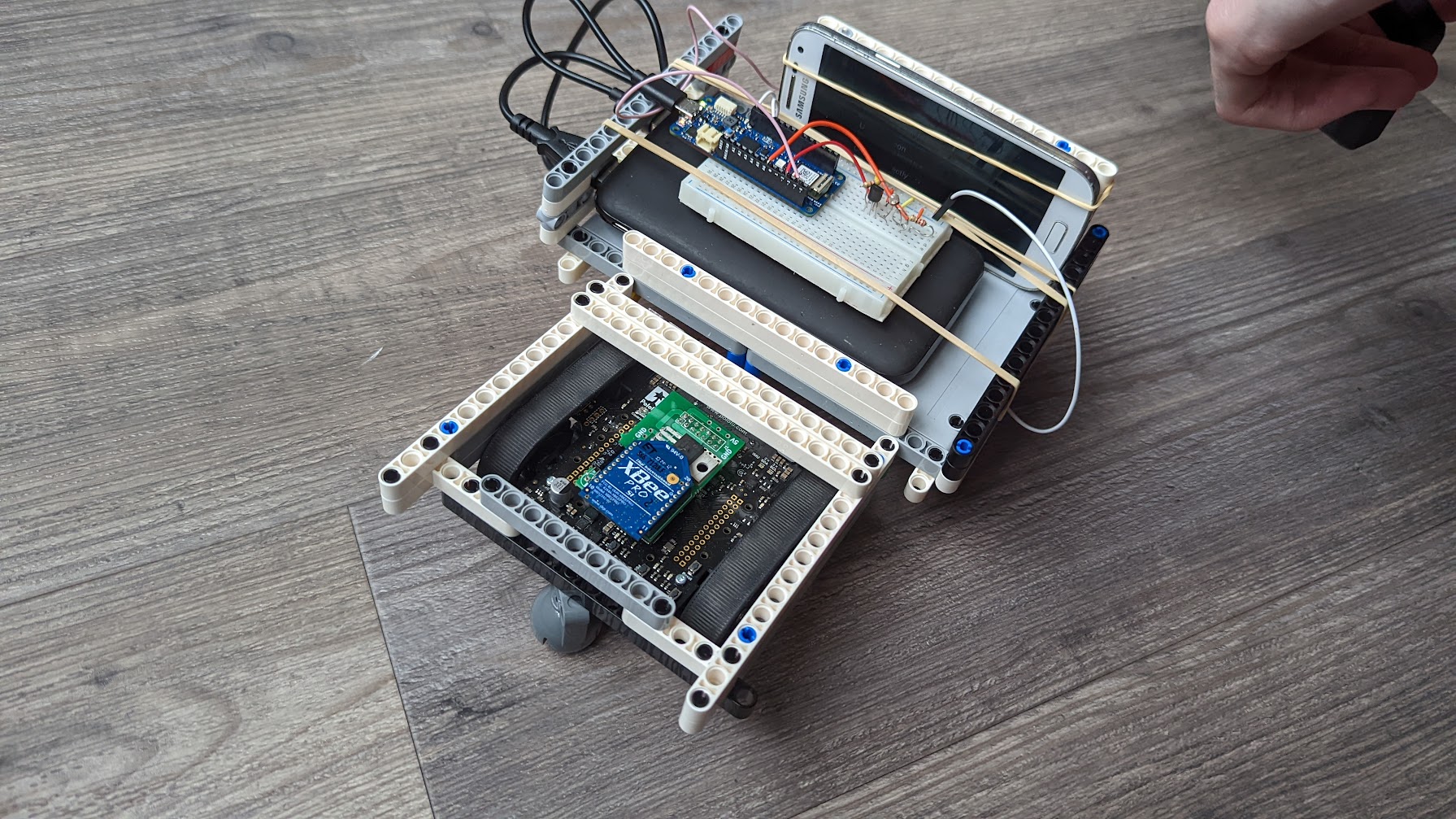


Figure 1: Final product

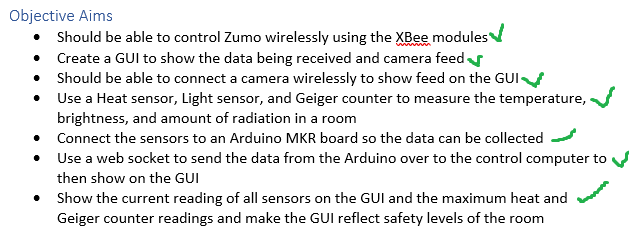


Figure 2: The Objectives set in out project spec

Our objective aims from the start of the project where all met, indicating that the project was a success.

# How did we do it

Throughout the development process we really wanted to highlight and discover new technologies we could put together to create something which would become symbiotic in the final form.

# Technologies used

## Arduino

The Arduino is an opensource device, which will allow us to create a microcontroller for the various sensors we need to read from.

The specific Arduino we used, the Arduino MKR 1010 Wi-Fi. Also HAS Wi-Fi capabilities, which we would rely on heavily for the creation of our project.

## Sensors

The sensors were the main catalysts of this project, as they are what read in the data from the outside world. The three variables we decided on detecting were: Heat, light, and radioactivity. These readings would allow our computer program to decided using complex algorithms, whether a certain zone is safe for humans or not.

## Heat

Heat being a strong indicator of a hazardous zone means that it is an obvious choice in our sensor setup

The heat sensor we used was the DS18B20 Digital temperature sensor, using this sensor and combining it with the libraries: OneWire and DallasTempreture, we could accurately find the temperature of the room in Celsius.

In figure 2 you can also see it fell within our needs in the project, measuring from “-55 to +125C” and not “requiring external components”.

Text

Description automatically generatedDiagram

Description automatically generated

Figure 3: The heat sensors pin configurations (Programmable Resolution, 2019)

A screenshot of a computer

Description automatically generated with medium confidenceText

Description automatically generated

Figure 4: The code that converts the signal to degrees C

By creating a new object of the new libraries, we could then run sensors.begin(), allowing us to then take readings from the pin 2, as seen defined in figure 2.

## Light

We incorporated light as a sensor as this could allow us to check for smoke or allow us to know if there Is power/sunlight within a room. This is especially important as we do not have night vision.

For the light sensor we used a photoresistor, this works by measuring the resistance in the current from the sensor, it works on an analog pin, meaning it works on giving us the resistance caused by the photoresistor, the higher the outputted voltage to the pin, the brighter the light must be.



Figure 5: reading the pin

As seen in figure 5, we simply can just take the value from the analog pin 6 and then send that data to then be processed through an algorithm to check if it is dark or light.

## Geiger Counter

The Geiger counter is what inspired us to create this project and then expended from this. In a scenario where there was nuclear war, or say a nuclear reactor meltdown, this device could tell a user if an area is safe to enter.

The Geiger counter works by conducting electricity through a gas filled glass tube when a high energy particle hits it. This then allows us to get a digital reading of how many counts per minute are happening at a given time, we then process this information when it is sent across to the server. Allowing us to judge the safety of an area.

A picture containing electronics, circuit

Description automatically generated

Figure 6: Geiger Counter

Programming of the Geiger was done in its own separate header file, keeping a clean file structure for the programming

Text

Description automatically generatedText

Description automatically generated

Figure 7:Geiger Counter Code

In figure 7 you can see the programming of the Geiger counter, when the sensor activates the function tube\_impulse() is run, with this we increase the number of counts, every log period this sets the new CPM by multiplying the counts by the result of 60/Logperiod. In theory you should be able to lower this to less then every 15 seconds, but in practice and after some investigation I found the Arduino sometimes is too slow for these calculations.

A picture containing text, electronics

Description automatically generated

Figure 8:Shows the Arduino Configuration

## WebSocket/Wi-Fi

We needed a way to wirelessly take the values from the Arduino and display them on a GUI/server. We used the Arduino MKR WIFI 1010, the Wi-Fi module on board the MKR allowed us to connect to the WIFI network. Thus, allowing us to send this information through a WebSocket. The Library WIFININA allowed us to connect to an internet connection.

Text

Description automatically generated

Figure 9:Connecting to the network

First, we used two-character arrays, one being the SSID and the other the password, these definitions were held securely in an Arduino.secrets file, allowing for more security when programming.

Text

Description automatically generated

Figure 10:Starting the connection to the server

Here we looped through the rest of the program whilst it was connected to the server, if it did disconnect, it would still continuously try to connect, allowing it to be fail/crash proof.

Text

Description automatically generated

The client is connected to the server by inputting the Ip address into the relevant char array. Then the WebSocket is created.

WebSocket allow for real time communication, they are used in any application where information needs to be sent quickly and then updated on the other end, for example a chat/messaging system. In our case we sent it to the GUI/server.

## Python/Tkinter

We choose to use python for the server-side operations, this is due to python having the appropriate libraries available for all connections needed to be made.

For the GUI side we used a library called Tkinter, this allows for a GUI to be created in python. Tkinter allowed us to display the values we needed as well as take the feed from the mobile phone display.

Graphical user interface, website

Description automatically generated

Figure 11: The GUI

The GUI contained color changing labels to indicate if the area was dangerous or not, we also logged the maximum values that the sensors gave, to then allow for better judgement.

Text

Description automatically generated

Figure 12: The webserver

The web server which was retrieving the information from the Arduino was setup on the python program, this was running as a thread alongside the GUI, the WebSocket was also running asynchronously. Meaning we could get the data before showing it.

Also hosted on the python program, was the serial port communication for the xbee to allow for the zumo to have wireless movement.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 13: Zumo control server Side

The code to control the zumo was also set up within a thread within the GUI application, the Serial library allows us to send data to the zumo, the zumo then knows how to process this information based on the commands given.

## Zumo

The zumo was simple to program as we already had created the code in the previous assignment, my code was taken and modified slightly to fit the new program.

The zumo only featured manual controls, which were WASD, these controls are easy to use and normal for this type of device. Meaning no learning curve should be needed.

## Phone camera

We originally were waiting for an Arduino camera, this never appeared for us. So, to improvise I took an old mobile phone and found an application that could send a live feed to a website. Allowing us to take this feed and implement it onto a GUI.

The phone is batty powered and is connected to Wi-Fi. The phone has its ports open for charging and has the capacity to go for multiple hours.

# Communication between technologies.

Chart, diagram

Description automatically generated

Figure 14:System design

In figure 11 you can see how all the technologies communicated, Between the Arduino and the sensors we used wires to connect the data, there is no other option for this.

From the Arduino to the GUI/Server we used a WebSocket to send real time information. The GUI sent information to the zumo through the xbee module, allowing us to send data through the serial port on the device.

The mobile phone first uploaded its real time image onto a webserver, we then took this image and were able to display it on the GUI.

# Issues faced

We faced many issues during the development of this device. To overcome theses challengers we faced, to overcome any issues we could use resources like google to succeed.

A big issue I faced was to do with the temperature sensor, originally I was using a analog temperature device, after facing some issues, I noticed the temp sensor would give skewed readings if The Arduino was connect to the Wi-Fi, after some research I discovered the Arduino could not use Wi-Fi and analog pins very arcuately at the same time, this was most likely due to power constraints or ever some interference in the wires.

To overcome this issue, I created a master slave configuration for the Arduino, this would separate the Wi-Fi and the temp readings to their own controllers.

We can send data from one Arduino to another using the sdl and sda pins on the device, as you can see in the following image.

A picture containing electronics, circuit

Description automatically generated

Figure 15Master slave config

Here we can see the master taking the readings from the analog pin and sending it to the slave device, which would then be able to process the data and send it to the server, over Wi-Fi.

In the end we did not use this configuration, as I found a digital temp sensor, which worked fine from the Arduino MKR, thus leading to all my efforts to be in vain.

Another big issue we ran into, was the movement of the device. The movement not being too reliable on certain surfaces. This was mainly due to the setup we had being rather top heavy, and not having a secure connection from the zumo to the frame of the Lego structure we created.

We tried many things to remedy this issue but concluded the structural integrity of the frame was at fault. To fix this in the future we could connect the zumo physically better, or even generate more power by adding a second zumo, meaning we have twice the motors to push the device.

# What I did.

Personally, In the project I worked on the Arduino side of things, First researching about WebSockets and Wi-Fi on the MKR to allow data to be sent wirelessly, then using this information to set up various forms of communication layers. implementation of all the sensors, the temperature, Geiger, and the light were done by me. I configured the code in a way that we could get accurate readings and then send these readings across the WebSocket in a timely manner. I made sure to even send the data in a readable format for the computer, I used JSON to allow the python program to be able to recognize what data values we have and allow the program to easily read what is being sent.

# What we could have done better.

In conclusion I think the project was overall a success, there were some factors like the movement of the device which could be improved upon, but I think in general we set out to fulfill the requirements set in our planning stage. We completed all out functional requirements and objectives we created, allowing for a device which can search a room for hazards and relay this information back to a user.

# Management strategies used.

We stuck to an agile development framework, we imposed various strategies to make sure we developed and created as a team.

The team had 1-week sprints where we would check the progress of one another, developing testing and talking amongst each other within these sprints.

We had a Kanban based system on Trello.

Graphical user interface, text, application

Description automatically generated