Final report

A picture containing text, sketch, drawing, diagram

Description automatically generated

29-06-2023

By

Jesper Bertelsen - au689481

Mads Jakobsen - au716748

Our initial plan

Before we began, we made a report about our plans for the project. In short, we wanted to make a home monitoring system for when you are out of the house. It would communicate through the internet using wi-fi to send data collected from the apartment. The data collected would include:

Light data collected using the light sensor to tell you if you forgot to switch off the light before leaving.

The gas sensor will notify you if the CO2 levels were too high.

The motion sensor will notify you in case of a break-in.

The temperature and humidity will be used for monitoring your plant in your home.

The keypad and the LCD would be used for the interface on the Arduino to make settings and monitor the data collected.

The buzzer will be used as an alarm.

Progress (and some bumps on the road)

We started out by making a plan for how we wanted to achieve our goals. The main issue was that an Arduino doesn’t have wi-fi. We would fix this by ordering a wi-fi module from imbedded stock. We also wanted to have some sort of an enclosure for the Arduino and all the attachments. This would be done by learning to use CAD software and 3D printing.

LCD

Early on we decided to use two resistors instead of a potentiometer. This way we are ensured that the LCD will always be setup properly and it won’t slowly drift. Also two resisters are easier to solder when we move away from a breadboard and assemble it.

Keypad issues

After having great success with some early testing of some of the sensors it was time to be humbled a little bit. We started working on the keypad and quickly realized that the Arduino only recognizes a keypress if the code is reading the state of the keypad the moment you press the key. In other words, if you press a key, and the code is not in the process of reading the input of the keypad, then the key won’t be registered.  
Our quick fix to this was simply to hold the button long enough for it to register. This was really annoying though, so we decided to find another solution.

The cause of the problem was a conflict of interest. We wanted the keypad to be updated instantly, but we also did not want all our sensors to check the temperature for example every few milliseconds. Afterall the temperature is somewhat stable in such a short timeframe.

After a lot of tinkering, we found a solution.

We started out by making a counter that would add 1 for every loop. Each sensor would be “hidden” behind an if statement that said if the counter was divisible by ‘x’ then it would check the sensor. This way we had some control over how often a given sensor is checked.

The Wi-fi module

The Wi-fi module caused a lot of problems for us. At first, we borrowed an ESP-8266 to use. This is a very simple module that only gives you Wi fi capabilities and nothing else. We tried following multiple guides on the internet with no luck. The problem was that the Arduino would not recognize that the module had been connected. We double checked all our connections with no luck.  
Then we tried reinstalling the firmware on the device. The thought was that, since it is a borrowed item, the previous user might have installed their own software on it. But even after attempting to updating the firmware we still had the same problem.  
At last, we decided to speak with the people at imbedded stock. After an hour of problem solving, we concluded that the device was faulty. They did not have any extra modules in stock, so we got an ESP-32 instead.

When we got the ESP-32 we got to know that some devices doesn’t use USB connections but UART connection. This was a breakthrough, as after downloading a USB - UART driver, we successfully flashed the ESP32.

The new ESP-32 however is not just a wi-fi module. It’s a microcontroller in itself. In other words, the ESP-32 is like an Arduino with a built in wi-fi antenna. This meant that there were no tutorials showing us how to send the data directly to the Arduino without using the ESP-32 as a microcontroller. We might have been able to use the ESP-32 by it self and by that replacing the Arduino. We didn’t know if we would be able to run that many modules through the Esp, not knowing if the ESP-32 would be powerful enough, so we decided to run them both, sharing data in-between them.

Through the Wire library, using the I2C connections, we successfully managed to decode the integers to an array of 3 bytes, 1 containing the key, indicating which sensor the data corresponds to, and 2 bytes containing the sensors value.

We were successful with encoding the data aswell, when we ran the data transfer separately from the rest of our code.

After encoding we then successfully send the sensor and it’s data to googles database, firebase, making it possible to see the data from anywhere in the world, when having internet connection.

Enclosure

We wanted to make a box for the Arduino along with the ESP-32, LCD, keypad and some other sensors. In order to do that we had to learn how to use CAD software. The university has free access to SolidWorks so that’s what we decided to use. Also in order to get access to 3D printers we needed a short course to learn how to use them.

The first design looked like this:

A drawing of a rectangular object

Description automatically generated with low confidence

Figure 1 Box version 1

A very simple box with some standoffs for the Arduino. There were no holes for cables to connect to outboard sensors. Another thing we quickly learned is that when you stare at a scaled-up model on your screen for a long time, 1 mm feels like a lot more than it is. The holes in each corner were meant for screws, but when it was printed they were too small.

A picture containing sketch, drawing, diagram, technical drawing

Description automatically generated

Figure 2 Box version 2

Version 2 had fixed these issues along with some other problems that made it hard to print. Additionally, the box was stretched to properly fit the Arduino.  
From here we made a bunch of renditions of the box with small improvements for every rendition.

A picture containing sketch, drawing, diagram, technical drawing

Description automatically generated

Figure 3 Box version 3

A picture containing diagram, rectangle, technical drawing, plan

Description automatically generated

Figure 4 Box version 3 top view

The last version has a lot of features added to it and the dimensions have changed. The biggest improvements are:

Keypad and LCD mounting hole

In the lid there has been made a cutout for the keypad and the LCD. On each side of the keypad there is a pillar on which a plastic stick is mounted across to keep the keypad in place. The LCD on the other hand is simply press fit.

Locking mechanism.

We ditched the screw holes for the lid and went with two hinges and a slider that keeps the box closed. The hinges are press fit in place. The locking mechanism uses a slider that fits into a hole in the box so it is kept closed. The slider has a spring behind it that keeps it from accidentally sliding back.

We learned a lot along the way about the dos and don’ts of CAD design and 3D printing.

Scenery

For the scenery, Jesper went to his father’s workshop, creating what’s supposed to be a miniature apartment, illustrating what our project is about. We then placed the sensors and installed its cables, and send out the data coming from them to the Arduino, hanging in the 3D printed cases, just outside of the scene.

The final setup

In the end we used all the sensors and devices along with the additional ESP-32. The ESP-32 is connected to the internet via wi-fi in order to send data to firebase where the user can monitor what is going on in the apartment.

The light, PIR and gas sensors were all mounted to the roof of our miniature apartment. They all send the information to firebase where the user can see if there is light, movement or smoke in the apartment. The gas sensor outputs a number and if that number is higher than 30 it will send the data to firebase and say there is smoke in the apartment.

The LCD will show the same data as on firebase but on the unit itself.

The keypad is for cycling between the different sensor inputs to view their data.

The buzzer is used as an alarm if the gas sensor is triggerd.

Conclusion

The project ended up not working as supposed to, when it all were put together. With our 3 processes in this project, 1. the data handling, 2. the I2C data transfer, 3. the cloud transfer, our code had grown enormous. We tried to simplify this by separating each segment into separate documents, which might have caused the extra problems, not knowing if we were pointing to the right way to the values. The data handling didn’t work correctly either, as our data didn’t change when doing the final run. Our cloud transfer did end up standing the test of time, working flawlessly since we first got it to work.

Even though the product ended up not working successfully, we learned how to conquer a lot of difficult obstacles, having it all work separately and we are sure, that with a few more days of debugging, the system would work as intended.

Successfully conquering the cloud transfer makes it possible, to in the future, be able to manipulate the data in ways, which could contain making graphical interfaces.