# ELEC 327 Final Project Report

Nonso Chukwurah ccc15
Github Link: https://github.com/TrippleCCC/MSP430-Home-System

## Project Description: Home Hub

The idea for this project was to make a home system that could monitor and report different information to the user via Bluetooth. Types of information include temperature, information about whether lights were on in the house, and information about if doors were opened or not. All this information would be bundled together in a data packet and sent to the user's phone via Bluetooth. This would happen periodically and the user would be able to access this information on their phone via an app. The user would also be able to send data back to the system. For example, the user could turn on a light in the home or maybe tell the system to stop sending data.

## Implementation

There are two main parts of this project. The first part is the implementation of the system on the msp430. The second part consists of implementing the user app.

### MSP430

Implementation of the msp430 was relatively simple. The code consists of multiple modules: a Bluetooth module, a data protocol module, and a temperature module. The Bluetooth module is mainly responsible for setting up UART communication. It also holds the interrupts for when we receive data from the user app. The data protocol module defines how we communicate data to the user app. It can take in a data packet and send it in a specific format over Bluetooth. The implementation of this will be discussed later in the report. The temperature module is simply used for getting temperature information from the on-chip temperature sensor.

### User App

The user app for this project is written in Flutter, a mobile UI framework for creating apps that can run both Android and IOS. So even though this app was developed on an android, I could potentially run on an IOS device as well! The layout of the app is also simple in nature. There are three main components: a Bluetooth module, a decoding module, and a module for the app UI and state management. The Bluetooth module is mainly responsible for receiving the data that we get from the msp430 and updating the state of the app. The decoding molecule takes the data

that we get over Bluetooth and decodes/translates it into a format that can be used in the app. The UI section of the app uses the previously mentioned modules and updates the UI as data changes.

## Data Protocol

The communication section of this app was probably the most difficult part. It can be understood by considering a unique situation: When the user app first makes a connection with the msp430 and starts reading data, how could it possibly understand the format of the data. The connection could potentially be taking place in the middle of a transmit so we might have some incomplete information. To remedy this I implemented a simple but useful data protocol. Essentially, before the msp430 sends over the data packet containing information about the sensors connected to the system, it sends over a series of 0s. After this, it then sends the actual packet. And finally, it ends the transmission by sending over another series of 0s. In addition to this general format, the data packet is structured in the following way.

```
struct __attribute__((__packed__)) data_packet{
    unsigned char device_status;
    long temp;
    unsigned char lightIsOn;
}; // Total bytes: 6
```

Notice the __attribute__((__packed__)). This makes it so that the compiler puts the elements in the struct in one contiguous piece of memory. This helps us when we are transmitting data because we won't get random 0s in between the elements of the struct. In addition, the device status is structured in a way such that it is never 0. So when we are checking the format of the data on the user app we are also checking to make sure that the beginning of the packet is not 0. Using this format it becomes way easier to detect data on the device.

# Reflection

Overall, I was able to get a prototype of the system working, both the msp430 part and the user app part. I was able to transmit information about the temperature sensor and the light sensor. In addition to transmitting the data, I was also able to send data back to the msp430 to toggle a light switch (this was mostly done for a proof of concept. With more time, I think I would be able to set up a led strip and change colors and patterns from the app). In terms of experience, I think I learned a lot about how to be efficient when sending data wirelessly. I didn't realize at first that I would have to be sending data byte by byte over Bluetooth and it definitely took me a while to understand how I would be able to implement the data protocol with this limitation.

I think that with more time and resources, I would be able to add more sensors to the system. An example of this would be an ultrasonic sensor that could monitor whether a door was open. This would be a sort of simple security system. I think I would have also been able to maybe transmit the data over wifi instead of Bluetooth. This would give me more options for implementing a user app (web app, desktop app, etc). Overall, I believe that my final product shows how capable a fully furnished system could be.

# Sources


http://elec327.github.io/final_project/
https://flutter.dev/