Dylan Danowski, Ryan Ciolek, Cole Cheman

12/19/22

Section 3

Embedded Systems

Github

Final Report: Weather Bug

	Dylan	**Ryan**	**Cole**
Initial Research	X	X	X
Block Diagram and UML	X	X	X
Analytical Calculations	X	X	X
PCB Design and Developing Booster Pack	X		
Writing Code	X		
Analyzing Result	X	X	
Research Related to the Project that Helps Report Writing		X	X
Writing Report	X	X	X

Abstract:

For our final project in this course we were tasked with creating an IoT application that is able to solve a specific problem of our choosing using at least 3 sensors. For our project, we have chosen to create, build, and test a weather monitoring system. For this system the 3 sensors that we have decided to use are a temperature/humidity senor and a photoresistor. Each of these sensors were installed onto a custom PCB that interfaces with the MSP430FR2355 microcontroller. The data that is collected from each of the sensors will then be sent to an ESP8266 WiFi module which will then send the data to a ThingSpeak application that will receive the data to be accessed anywhere.

Introduction:

One of the biggest evolving areas of technology are IoT's (Internet of Things) which are being used to create smart systems that make things in our everyday life easier. These things range from creating a smart home security system all the way to monitoring the current weather outside in a given area which is the project that we have chosen to create. We are calling it the "Weather Bug" weather monitoring system.

The system uses 3 sensors, temperature, humidity, and a photoresistor to detect sunlight. We tried to successfully interface our sensors to the MSP430 but we ultimately opted to go with an ESP32 arduino. Our reasoning as to why we chose this route will be explained in the "Experimental Setup and Results" section. Through this method we were successfully able to get the sensors to read data and relay that data to us.

Background:

In order for us to successfully create an IoT we had to break down what we needed to do in order to accomplish our task. After we grouped up we then had to come up with an idea of what we wanted to do and what sensors we needed to do it. We settled on creating a weather detection system and determined that we would need 3 sensors, one that could detect various light levels in the environment, one that would be able to detect the temperature, and one the would be able to detect how humid it was in the given area. Once we found our temperature/humity sensor and photoresistor that we wanted to use we then had to design a custom PCB that would be able to communicate with the MSP430. However as stated above we opted to use a ESP32 arduino in which case we utilized a breadboard to successfully connect all of our sensors and interface with it. Since we don't have a lot of components in this particular project we were able to recreate what we would do on a PCB to a breadboard. Once we had our prototype done of our project we then had to demonstrate the functionality of our design with all of its features and what data it could gather.

Methodology and System Design:

The following is a Level 1 Block Diagram depicting the data flow of the WeatherBug:

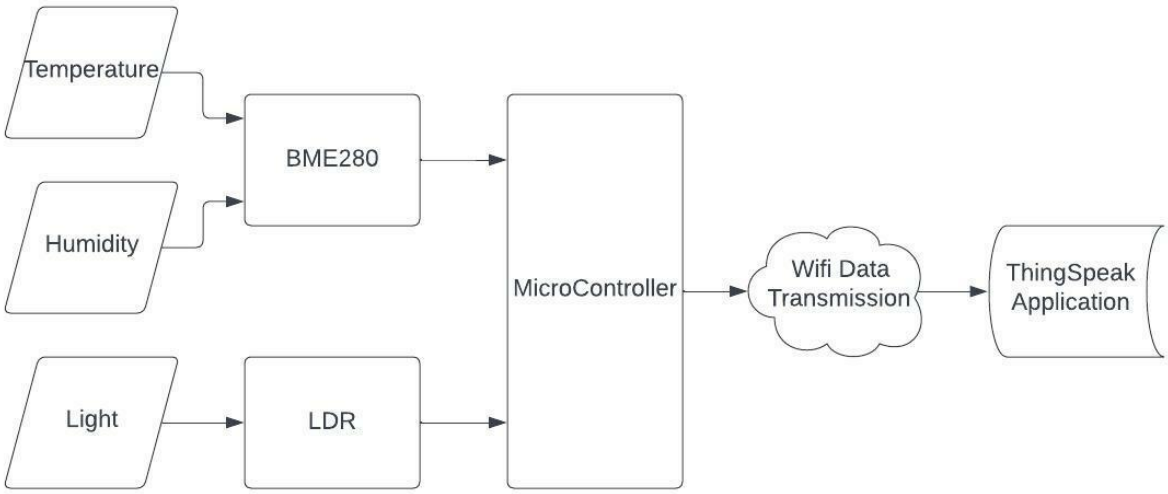

Figure 1: Level 1 Block Diagram of Weather Bug

The WeatherBug takes in three different measurements from the local environment: it reads temperature, humidity, and light. The temperature and humidity are translated by the sensors of the BME280, and sent to the microcontroller via an I2C communication bus. The light level read by the photoresistor (LDR) changes its resistance. Using a simple voltage divider circuit and a peripheral analog to digital converter, the microcontroller can use the change in voltage induced by the LDR to determine the relative light level in the surroundings.

The below figure is a UML diagram outlining the algorithm used by the microcontroller to systematically collect and send data.

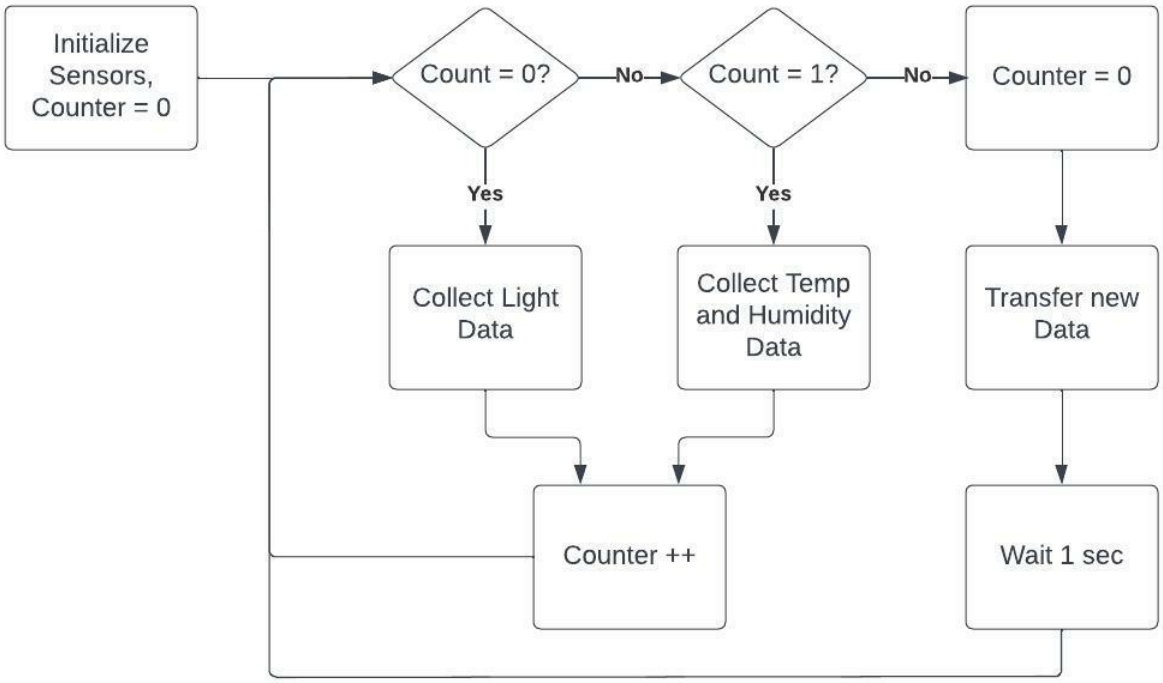

Figure 2: UML Diagram of Microcontroller

The microcontroller uses a counter to run one of three functions. In Mode 0, the relative light level (light/dark) is collected from the LDR. In Mode 1, new temperature and humidity data is collected from the BME280 and stored in a register. In Mode 2, the data from Modes 0 and 1 are transferred via Wifi to the ThingSpeak application, the counter is reset, and there is a one second duration until the next wave of data collection takes place.

Experimental Setup and Results:

In order to ensure that all of the Weather Bug was working properly, each of the sensors were individually tested. Due to the MSP430's incompatibility and lack of documentation, an ESP32 was used because of its ability to interact with the sensors using Arduino. First, the light sensor was designed and run. It was tested by changing the lights within an environment and seeing how the sensor would change. The sensor was set in a bright room initially. Then, the lights were suddenly turned off to see if they would react. As shown in the graph below, the sensor's light level suddenly drops on the graph at the same time the light in the room shuts off.

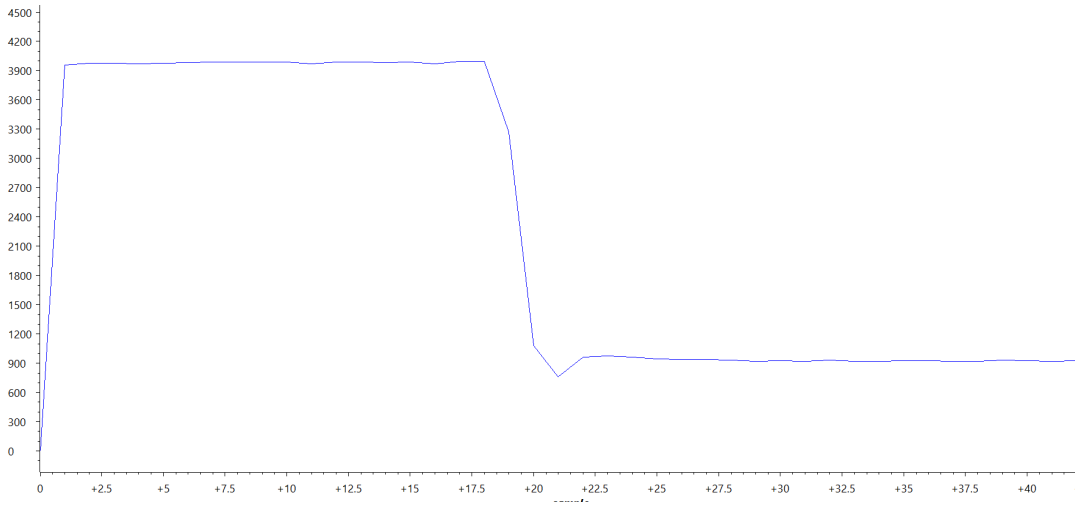

Figure 3: Time vs Light Sensor

Next, the temperature and humidity sensors on the BME280 were tested. Initially, the BME280 was supposed to interact with the MSP430 by using two of its I2C pins. At first, by following the BME280's datasheet, we attempted to program the board and collect the data. Initially, it seemed reasonable after we identified and fixed a few obvious problems that we had neglected early on. Still, after days of trying and failing to receive data using I2C, we decided to switch our method to something that has been much more documented. By using Arduino code to interact with a ESP32 instead of the MSP430, the data was able to be collected and read back accurately. An example of this data can be found in the figure below.

ESP32 with BME280

MEASUREMENT	VALUE
Temp. Celsius	22.99 *C
Temp. Fahrenheit	73.38 *F
Pressure	1027.14 hPa
Approx. Altitude	-115.02 m
Humidity	56.43 %

Figure 4: BME280 Values using ESP32

Conclusion/Future Work:

In conclusion, this project taught us how to use sensors and embedded systems to design and program a PCB, collect and provide data, interact with other embedded systems. One of the hardest parts with this project was being able to interact with the temperature and humidity sensors in the BME280. By using both our resources and knowledge to our advantage, the ESP32 was able to interact with the BME280 and provide the data that was preventing the final project from being completed. This piece was specifically very rough because of the lack of Windows documentation with the BME280 and the MSP430. By using Arduino which is heavily documented, it became a much more feasible task though. In the future, it would be very useful to find a way to reattempt this problem in order to collect data faster and more accurately.