Final Project: Smartish-Home

Jeremy Beal, Andrew Mazurek and Cory Tindall
Embedded Systems Section 4
12/19/2022

Contribution Chart:

	Jeremy Beal	Andrew Mazurek	Cory Tindall
Initial Research	X	X	X
Block Diagram & UML	X	X	X
PCB Design	X		
Soldering	X	X	
Coding	X		
Analyzing Results	X	X	X
Research related to the project for Report	X	X	X
Writing Report	X	X	X
Making Presentation Slides	X	X	X

Abstract

To demonstrate the concepts and ideas learned in this course, students were tasked to create an Internet of Things application that involves at least three sensors and MSP430 interfacing to ThingSpeak. For this project, the team decided to base it around the “Smart Home”. This Smart Home project uses a temperature sensor, motion sensor, leak sensor and a LCD display. These sensors were implemented into a PCB design that would ultimately connect to the MSP430 microcontroller board in order to execute their applications. Once all the pieces are connected together, the Smart home system is therefore created. Each of the three sensors have a distinct and important function to the system. The temperature sensor paired with the LCD display represents a smart thermostat system. The motion sensor paired with a lightbulb displays a smart lighting system. Lastly, the leak sensor paired with a buzzer shows the basement protection system. The data collected from these sensors are transmitted to the ThingSpeak cloud through the Code Composer Studio using C and C#. All three of these together represent the smart home embedded system.

Introduction

With the constant development in technology, once rare capabilities are becoming more common. One of these examples is the Internet of Things, or IoT devices. Even though the name IoT may not be a commonplace, these devices can be seen in many instances, especially in our homes. Referred to as “smart devices” these devices include Amazon’s Alexa, Ring, and the Samsung Smart Fridge. All of these devices receive an input whether its voice activation or motion sensing, and communicate an output to the user via the internet, text messaging, or its own web application. The Smartish Home project is an IoT device that uses ThingSpeak in order to communicate the output of three different sensors. The sensors include the internal MSP430 temperature sensor, the EKMB1301113K motion sensor, and the 101020018 leakage sensor. A NHD-0216K1Z LCD display was used to show the outputs of all three sensors. The buzzer was also used to show that the leakage sensor works by activating when water is detected. A lightbulb was also connected to the motion sensor and relay that turns on when motion is detected. The sensor data values are periodically converted to strings and transmitted serially through UART. A C# program takes the transmitted string and breaks it into the three values for each of the three sensors. These values are sent to their respective fields on the ThingSpeak web platform via API key and channel ID. With the Smartish Home fully together, it has the potential to be put in every home to help improve the daily lives of its users. Not many houses contain sensors that can detect leaks, which can lead to costly damages and repairs. The Smartish Home can help recognize these leaks in combination with improving power consumption in order to help protect the family in the home.

Background

In order to begin this project, a team was formed and the final project idea was developed. This idea was chosen from a list of ideas including smart home, smart car, live weather modeling and smart gardening system. The next step was to select at least three sensors that would each have an application involved in the project idea. From there a PCB would be designed to incorporate all of the sensors and the MSP430 microcontroller into the smart project embedded system. In order to do so, the software DipTrace was used to create the schematic and model for the PCB. The PCB was to be designed in the most efficient way possible by using as little space as possible while still being able to function properly. Once the PCB design was checked, the Gerber files would be sent out so the board could be printed. Also a Bill of Materials would be created for the required PCB parts. Once the PCB and parts arrive, the system would then be assembled by soldering the parts onto the board. The chosen sensors would then be connected to the board and a code would be developed in order to have the system and sensors run properly. Lastly, the smart system would be tested by uploading the code onto the MSP430 board and running it through the created system, testing each component. The functionality of the system would then be presented and any issues with it would be discussed.

Methodology and System Design

With the chosen topic of “smart home”, the first thing the team had to discuss was the functionality of the desired embedded system that was to be created. It was decided that the project would be centered on three aspects of a house that can be improved upon to be considered a “smart home”. These aspects include a smart thermostat, smart lighting system and smart basement protection system. Figure 1 demonstrates the basics of how these sensors interact to create the “smart home”.

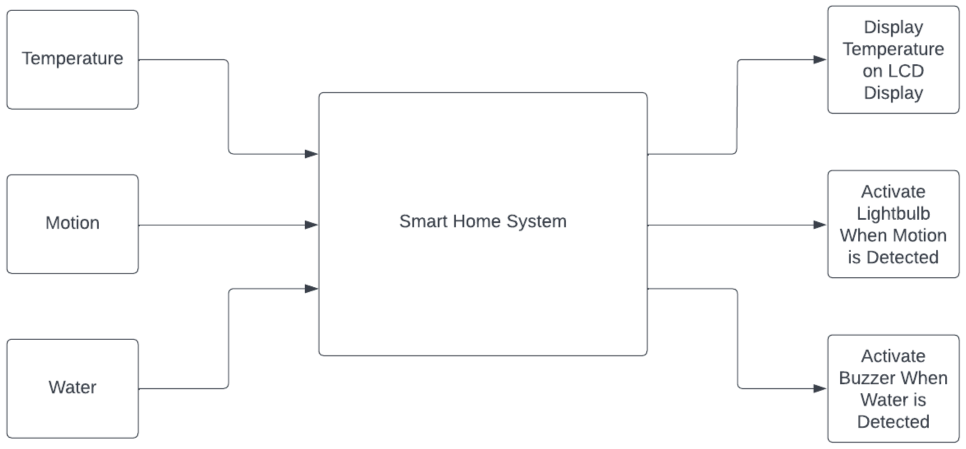

Temperature
Motion
Water
Smart Home System
Display Temperature on LCD Display
Activate Lightbulb When Motion is Detected
Activate Buzzer When Water is Detected

Figure 1: Level-0 Block Diagram

The Level-0 Block Diagram was the first step in the design process. Once the idea of the Smartish-Home and the sensors that would be used were established, a way to map out the inputs and outputs of the system had to be developed. Figure 1 does just this as the smart home receives three inputs and outputs their respective results. The temperature, motion, and water are all detected and then relayed to the microcontroller. This data will then be output on the LCD along with being in ThingSpeak. This includes the temperature, if there was water present, and if there was motion detected. In order to fully understand the idea of the system, a more specific block diagram was created. Figure 2 shows the Level-1 Block Diagram for the Smartish-Home.

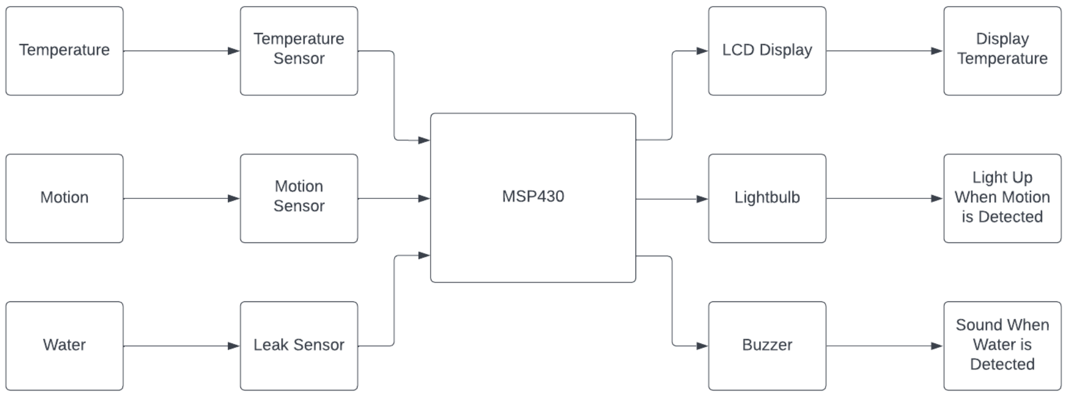


Figure 2: Level-1 Block Diagram

Once a basic understanding of the project was achieved, the Level-1 Block Diagram was created to dive deeper into the design process. The inputs and outputs of the system are the same except with an extra layer on each. The temperature, motion and water will be detected by their corresponding sensor before being imputed into the MSP430 board. The temperature will be read by the internal temperature sensor of the MSP430. The EKMB1301113K motion sensor will be used to detect motion and the 101020018 leakage sensor will be used to detect the water. The data from these three sensors will then be run through the MSP430 and different outputs will occur according to the input. The value of the temperature will be displayed by using an LCD display. A lightbulb will light up when motion is detected and a buzzer will sound when water is detected. The data from these three sensors will also be outputted to ThingSpeak. Now that the idea of the project is mapped out, the next step was to create an UML diagram to represent the programming that goes along with the system. Below in Figure 3 is the UML diagram.

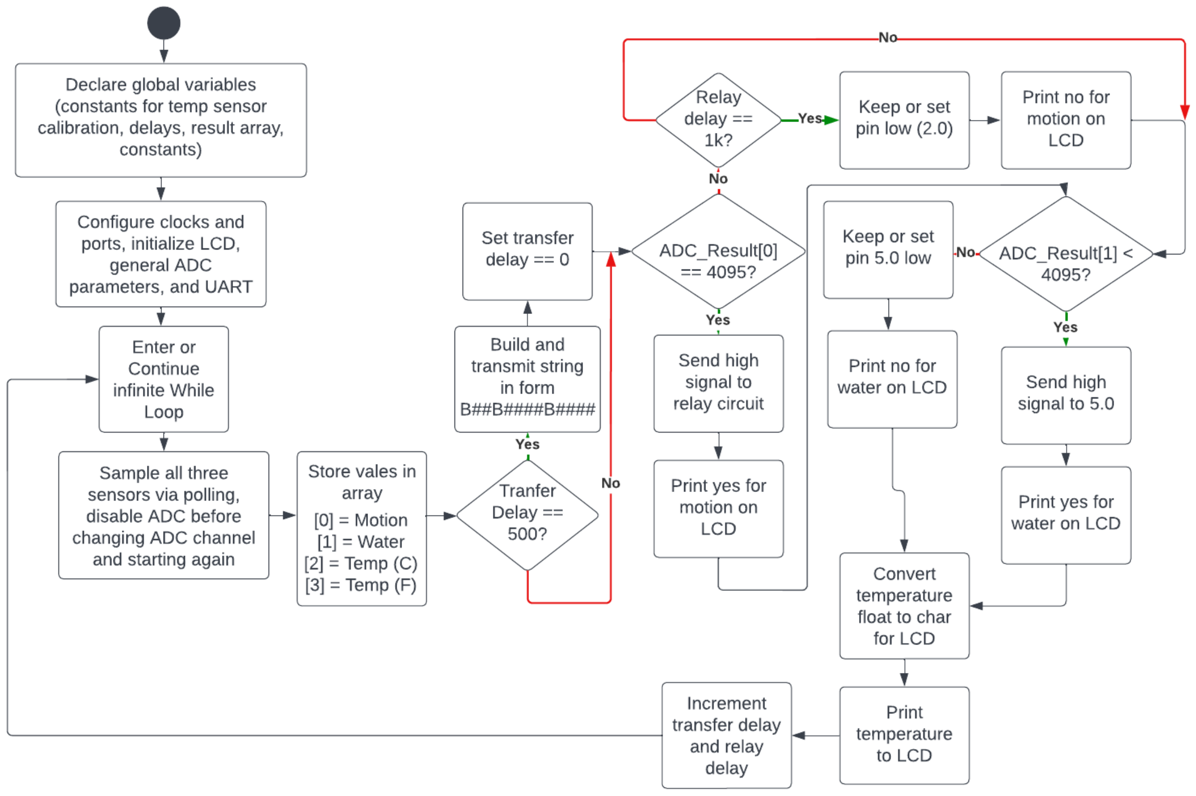
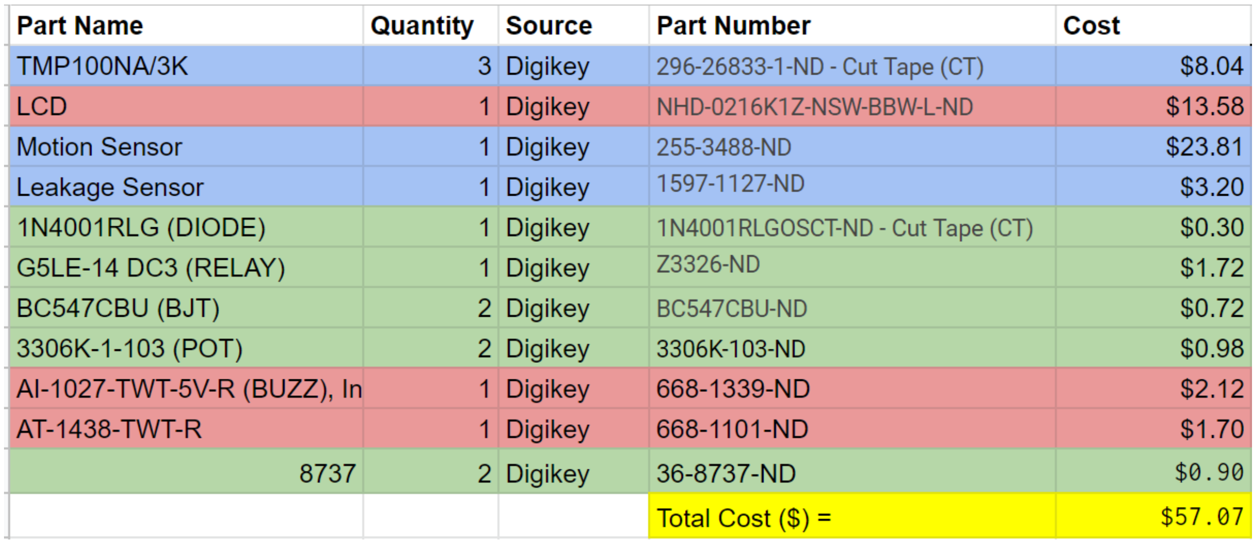

Declare global variables (constants for temp sensor calibration, delays, result array, constants)
Configure clocks and ports, initialize LCD, general ADC parameters, and UART
Enter or Continue infinite While Loop
Sample all three sensors via polling, disable ADC before changing ADC channel and starting again
Store vales in array [0] = Motion [1] = Water [2] = Temp (C) [3] = Temp (F)
Tranfer Delay == 500?
Yes
No
Build and transmit string in form B##B####B####
Set transfer delay == 0
ADC_Result[0] == 4095?
Yes
No
Send high signal to relay circuit
Print yes for motion on LCD
Relay delay == 1k?
Yes
No
Keep or set pin low (2.0)
Print no for motion on LCD
ADC_Result[1] < 4095?
Yes
No
Keep or set pin 5.0 low
Print no for water on LCD
Send high signal to 5.0
Print yes for water on LCD
Convert temperature float to char for LCD
Print temperature to LCD
Increment transfer delay and relay delay

Figure 3: UML Diagram

The above UML diagram shows the flow of the code. First, the clocks and ports are configured before initializing the LCD, general ADC settings, and UART. The program then goes into an infinite loop. To get the ADC values, the correct ADC channel must be selected. The ADC channel selection can only be done when the ADC is disabled. The system waits until the ADC conversion is complete (polling) before moving onto the next (where ADC is disabled and channel is changed). The data needs to be transmitted to ThingSpeak, but this does not continuously happen. This only happens when a variable counts up to 500. The string is built using ‘B’ as the character split point, a ‘B’ is added after each sensor data value. After that the transfer_delay variable is reset to 0. There are a few ‘if’, ‘else’ conditions that the program goes through. The first concerns the relay circuit. If motion is detected (based on the value stored in the results array), a high signal is sent to the relay circuitry to trigger it. The relay_delay variable is used to keep the relay active so that it does not flicker rapidly. The relay will turn off if the relay_delay variable is equal to 1000 and the motion value is less than 4095. The relay_delay variable is also reset to 0. The temperature value from the internal temperature sensor is converted from float to string for output onto the LCD display. General strings are used for the water and motion fields (which will either be yes or no depending on the logic described above). This process repeats itself as the loop continues. For getting data to ThingSpeak, a C# program is used. The transmitted string from CCS is received by the C# program. The string is split based on the location of the 'B's. The separate sensor values are stored in an array and are uploaded to the respective ThingSpeak field from there. Figure 4 below shows the appropriate parts needed in order to assemble the Smartish-Home system.



Part Name	Quantity	Source	Part Number	Cost
TMP100NA/3K	3	Digikey	296-26833-1-ND - Cut Tape (CT)	$8.04
LCD	1	Digikey	NHD-0216K1Z-NSW-BBW-L-ND	$13.58
Motion Sensor	1	Digikey	255-3488-ND	$23.81
Leakage Sensor	1	Digikey	1597-1127-ND	$3.20
1N4001RLG (DIODE)	1	Digikey	1N4001RLGOSCT-ND - Cut Tape (CT)	$0.30
G5LE-14 DC3 (RELAY)	1	Digikey	Z3326-ND	$1.72
BC547CBU (BJT)	2	Digikey	BC547CBU-ND	$0.72
3306K-1-103 (POT)	2	Digikey	3306K-103-ND	$0.98
AI-1027-TWT-5V-R (BUZZ), In	1	Digikey	668-1339-ND	$2.12
AT-1438-TWT-R	1	Digikey	668-1101-ND	$1.70
8737	2	Digikey	36-8737-ND	$0.90
			Total Cost ($) =	$57.07

Figure 4: Design Cost of Smartish Home

Figure 4 is the Bill of Materials for the Smartish-Home. Each part is individually labeled with the number of parts used and the total price of the parts. The bill is broken down into three sections: sensors (blue), functionality pieces (green), and output components (red). At the end of the planning stage, there are 11 different parts and 16 total parts. The combined cost came out to be $57.07. Once the bill of materials was submitted, the next step in this project was to design and construct our PCB in order to test our final results.

Experimental Setup and Results

The first steps in creating the PCB was to design the schematic for the Smartish-Home. Once the schematic was completed, the PCB was able to be constructed and sent out for manufacturing. When the board was received, the PCB was then soldered with the parts from the parts list to perform basic testing. After testing the individual sensors, C# was used to communicate with both the LCD display and ThingSpeak to verify the results. Figure 5 shows the full schematic file of the project. Portions of this schematic will be gone into in more detail shortly. It is worth noting that the temperature sensor portion was not implemented as the on board MSP430 temperature sensor was used instead. Additionally, a spot for a WiFi module was included but was not implemented since the data transmission was hard wired.

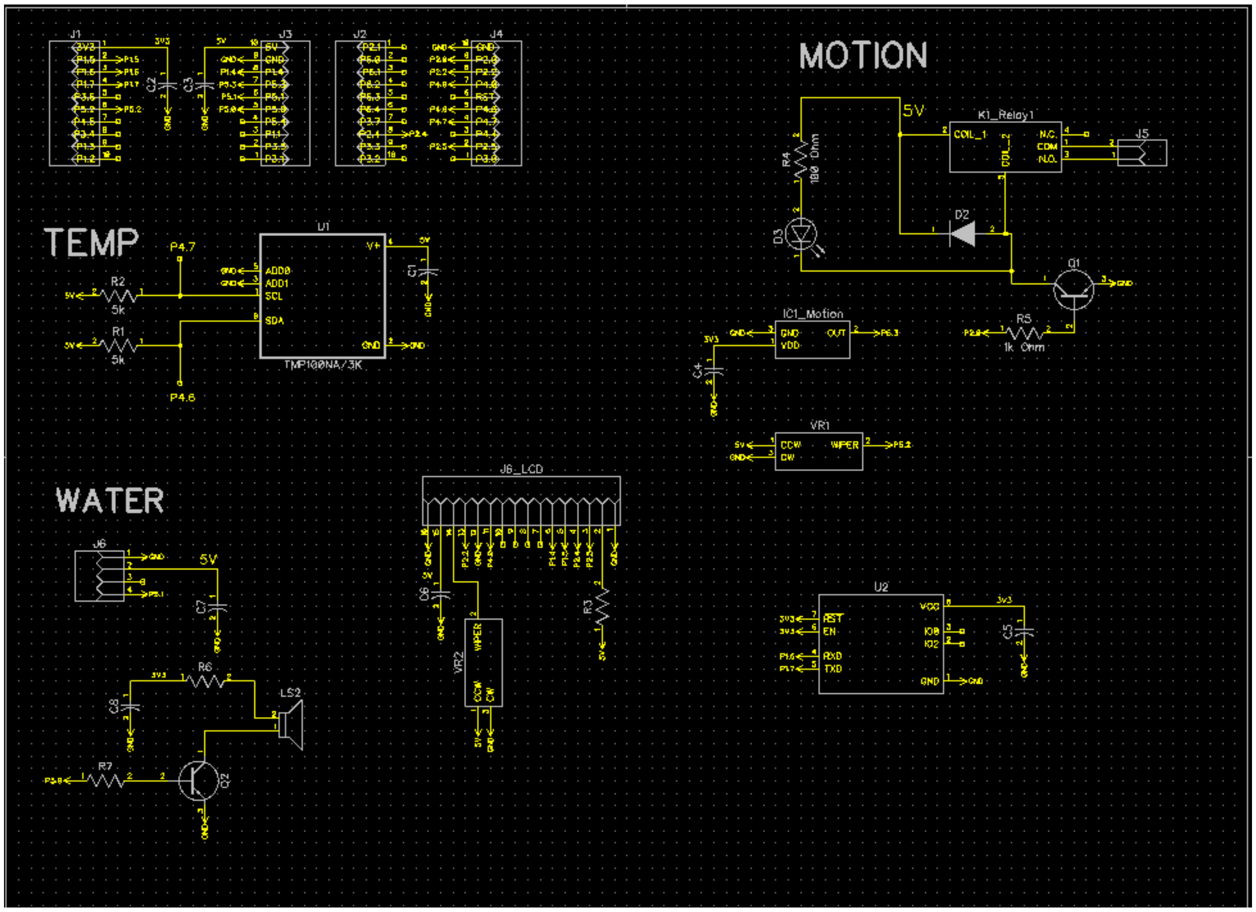


Figure 5: Smartish-Home Schematic

The schematic in Fig. 5 breaks down each connection for the three sensors to the MSP430. It also includes how each of the parts from the bill of materials connects to each sensor in order for them to function properly. The following figures show the schematic diagrams of each individual component and will talk about how each sensor functions as it is connected to the MSP430 microcontroller. Figure 6 shows the schematic for the motion sensor.

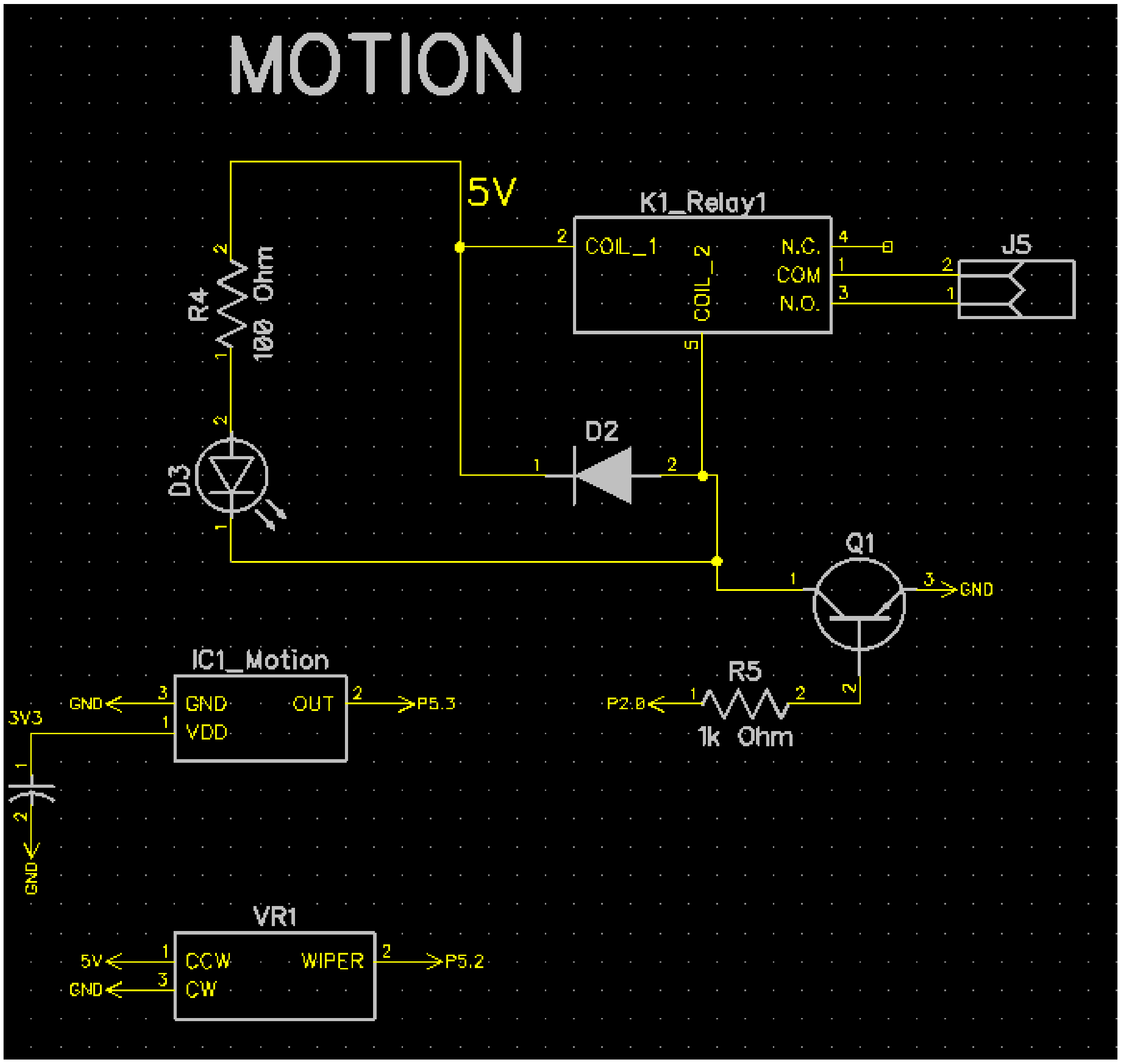


Figure 6: Schematic for the Motion Sensor

The digital motion sensor has two inputs, ground and VDD (which is connected to 3V3). The output of the motion sensor goes directly into P5.3 which includes an ADC channel (channel 11). After the input into the MCU is sampled, logic within the code decides whether there is motion or not and sends a signal based on that. If there is, a high signal is sent through P2.0 which is connected to the base of a BJT NPN transistor. This high-side switch has its emitter connected to ground while the collector is connected to one of the relay coils. There is a flyback diode between the two coils for protection (spikes in voltage). There is an LED in place between the two coils that verifies that the relay is triggered. Only the normally open and common ports of the relay will be used. The side of the coil that isn't directly connected to the transistor is connected to 3V3 (Schematic shows 5V but that relay was replaced with a 3V3 version since the MSP430 is a 3V3 board). Additionally, there is a potentiometer that was planned to be used as a sensitivity adjuster but it was not implemented in code. The schematic for the next sensor (the leakage sensor) is shown below in Figure 7.

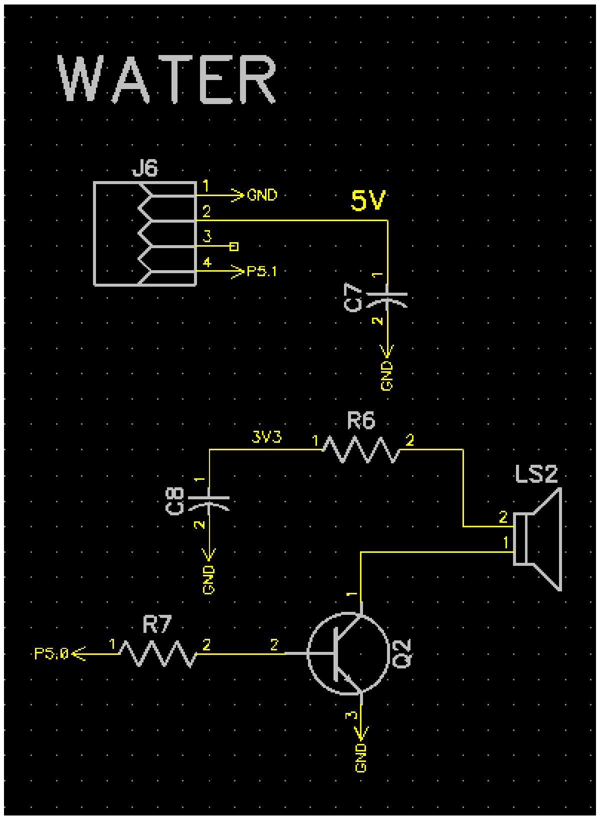


Figure 7: Schematic for the Water Sensor

The water sensor schematic is similar to the motion sensor / relay schematic. There are three connections made to the water sensor, ground, VCC (connected to 5V), and a signal output. The output depends on the state of exposed PCB traces. If the traces are left alone, a resistor will pull the sensor trace value high. If the traces are submerged in water, the PCB traces will be shorted resulting in a lower signal output value. This information is fed directly to P5.1 (ADC channel 9). Based on the value from the ADC sample, P5.0 will be set high or low. So, when the sensor encounters water, P5.0 will be driven high into the base of another NPN transistor (similar to the relay circuitry). The emitter of the transistor is connected to ground while the collector is connected to one side of an externally driven buzzer. Since the buzzer is externally driven it only requires a DC voltage to operate, the other side of the buzzer is connected directly to the ground. Shown below in figure 8 is the PCB layout from the schematic.

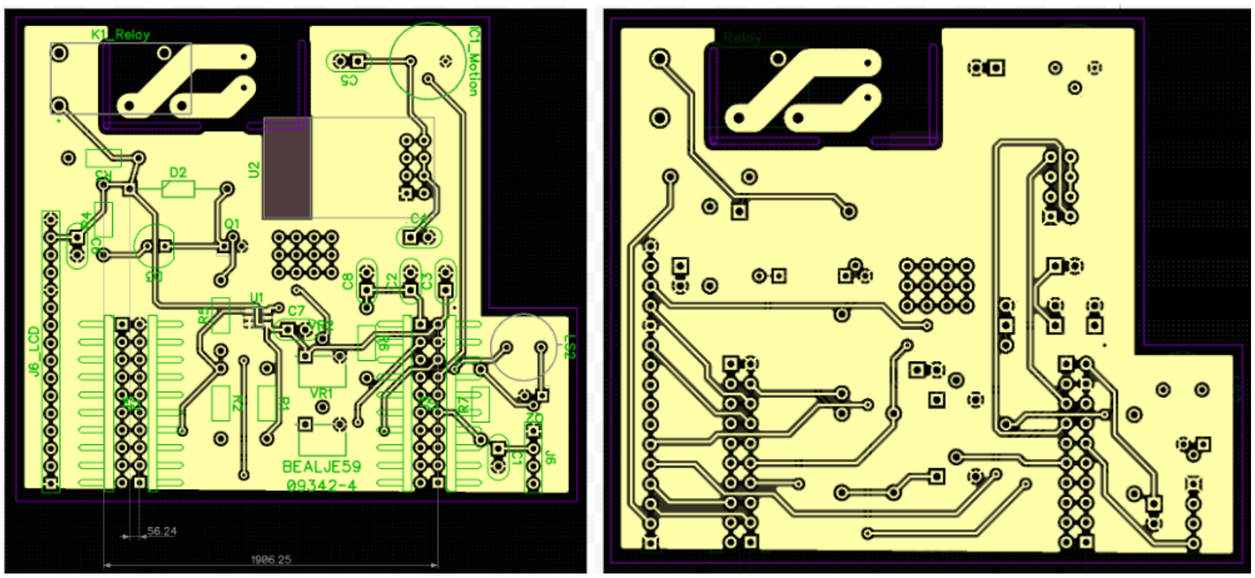


Figure 8: PCB Layout

The PCB layout is straightforward. The most important part was getting the spacing of the headers correct. Additionally, the capacitors needed to be placed in the correct locations (10 uF before any item gets power, 1uF before specified component gets power). Since a portion of the relay is high power, the traces needed to be adjusted so that they could handle the increased current. These thick (140 mils) traces were added on the top and bottom of the board. There were also board cutouts placed to separate the low power section from the high power section. Once the PCB was ordered, the next steps were to solder the board once it arrived. Figure 9 displays the completed setup of the Smartish-Home.

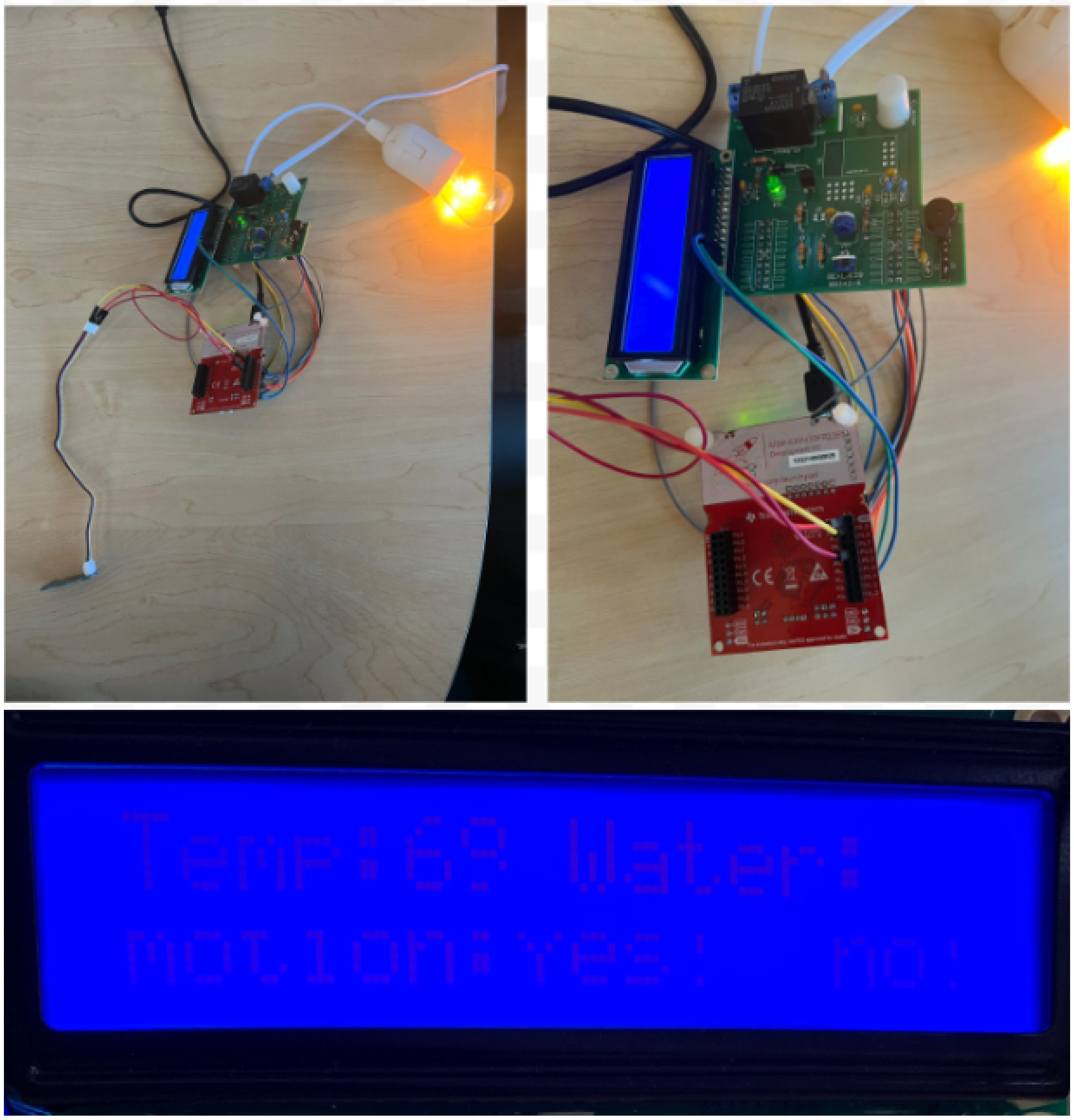


Figure 9: Soldered PCB Connected with MSP430

Fig. 9 has all of the parts soldered in along with the wire connections to the MSP430. The lightbulb that is used as the motion sensor output is manually connected to the relay (the hot wire is split in two, one end goes into the common port, the other goes into the normally open port), which tells it when to turn on through Pin 2.0. The leakage sensor is connected to Pin 5.0 in order to set off the buzzer whenever the traces on the sensor connect. The LCD display is connected to four data pins to show the outputs, two pins (power and ground) for power, and a connection to a potentiometer for screen contrast. When powered through a USB connection to the laptop, Smartish-Home is fully powered and is able to display all of the desired outputs on the LCD display. With the setup complete, the next steps were to test the board by communicating through ThingSpeak. The resulting data collected from the three sensors is shown below in Figure 10.

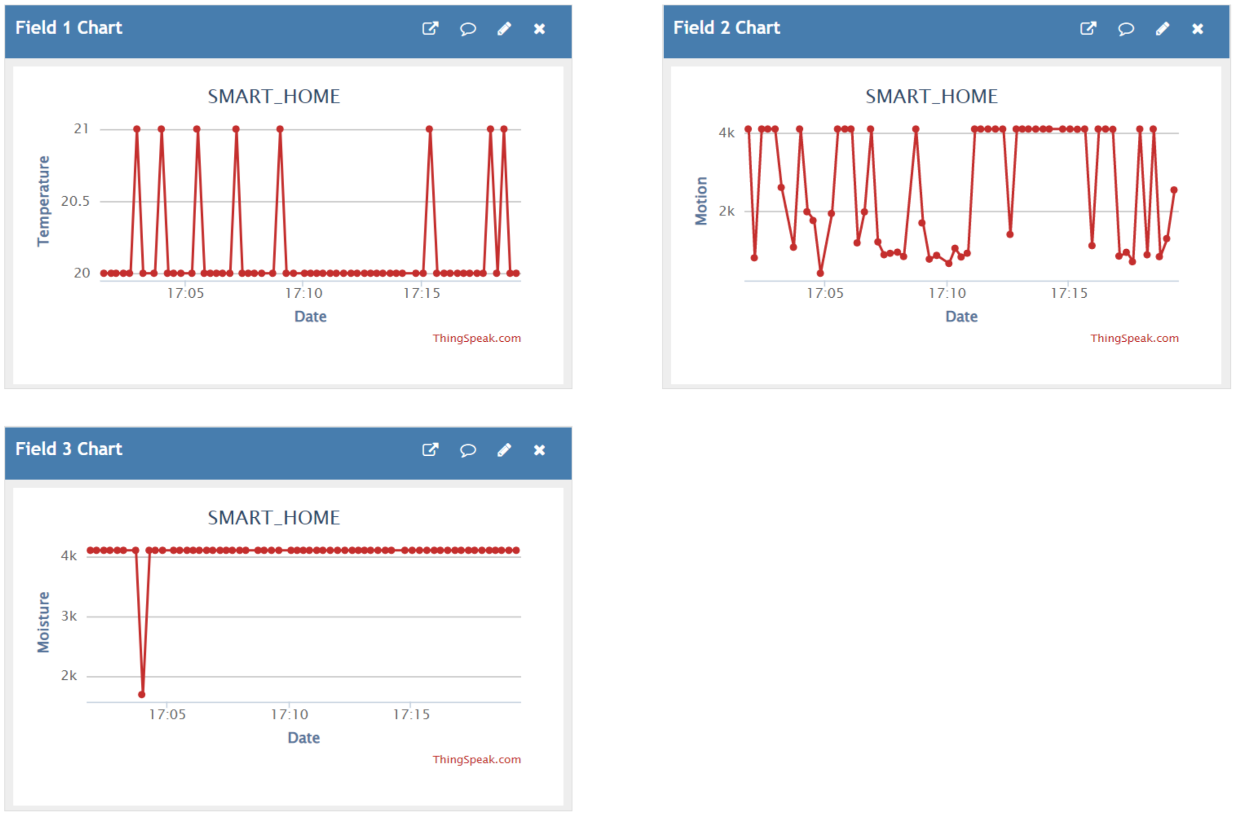


Figure 10: ThingSpeak Results

As shown, the graphs display the data from the temperature, motion, and leak sensors. The ThingSpeak graph for the temperature sensor (Field Chart 1) simply displays the temperature of the surrounding area every minute. This temperature value will also be displayed on an LCD. Field Chart 2 depicts the motion sensor activity. When the value is at 4095, the motion sensor detects motion and the signal is sent to turn on the light bulb. The LCD will then display “Yes”. If a value below 4095 is detected, there is no motion and the light bulb will shut off. The LCD will display “No” under motion. Lastly, Field Chart 3 represents whether or not moisture is detected by the leak sensor. Here, the points at 4095 on the graph represent when there isn’t moisture. When moisture is detected, a buzzer will sound and the LCD will display “Yes” under the word water. If no moisture is detected, the buzzer will turn off and the LCD will display “No”. The value on the ‘Field 3’ chart will also be visibly lower.

Conclusions/Future Work

Through the completion of this project, a working IoT device was able to be constructed. However, that does not mean there is room for improvement. The temperature sensor that was incorporated into the project only recorded temperatures, it did not act on the values it provided. With the success of the relay circuit for the motion sensor, an additional circuit could be created that allows the activation of some high power item (i.e fan, air conditioner) based on the temperature value. This would turn a simple temperature monitoring system into a true smart thermostat. Additionally, the timing on the relay circuit could be improved with added functionality to control sensitivity (as of now the only way is through changing the relay_delay variable in code). The project was originally planned to include a WiFi module, however that was not implemented. The best version of our system would not be dependent on an external computer, so a system that could work independently (i.e. MSP430 programmed independently with WiFi module) would be a logical next step for this project.

Creating Smartish-Home allows the group to understand the development of IoT devices. Being able to choose our own parts and use them to create a final product helped improve some critical skills that ECE’s need to succeed out in the field. After testing the final design, the group was successful in getting three different sensors to communicate with ThingSpeak. The LCD was able to verify this communication line by displaying the outputs that aligned with the ThingSpeak graphs. As a result, a working “smart home” device was able to be created that can control lighting through motion, actively display the temperature, and set off an alarm if liquid is detected in the home. Overall, not many houses in today’s world have smart capabilities. Smartish-Home provides these services to the homeowners to help improve the quality of life and has the potential to protect the home that it's working in. With the improvements mentioned above, Smartish-Home has the possibility to be implemented in every home.

References

[1] “What is the Internet of Things (IoT)?,” *Oracle.com*, 2022. https://www.oracle.com/internet-of-things/what-is-iot/ (accessed Sep. 14, 2022).

[2] “MSP430FR2355,” *MSP430FR2355 data sheet, product information and support | TI.com*. [Online]. Available: https://www.ti.com/product/MSP430FR2355. [Accessed: 4-Oct-2022].

[3] Baldengineer, “Low side vs. high side transistor switch,” *Bald Engineer*, 17-Apr-2019. [Online]. Available: https://www.baldengineer.com/low-side-vs-high-side-transistor-switch.html. [Accessed: 4-Oct-2022].

[4] Robojax, “Control AC bulb with 5V relay using Arduino,” *YouTube*. Aug. 22, 2019. Accessed: Dec. 19, 2022. [YouTube Video]. Available: https://www.youtube.com/watch?v=g6k8sPJyif8

Appendix

GitHub Link: https://github.com/jerbo2/EMBEDDED_FINAL_PROJECT