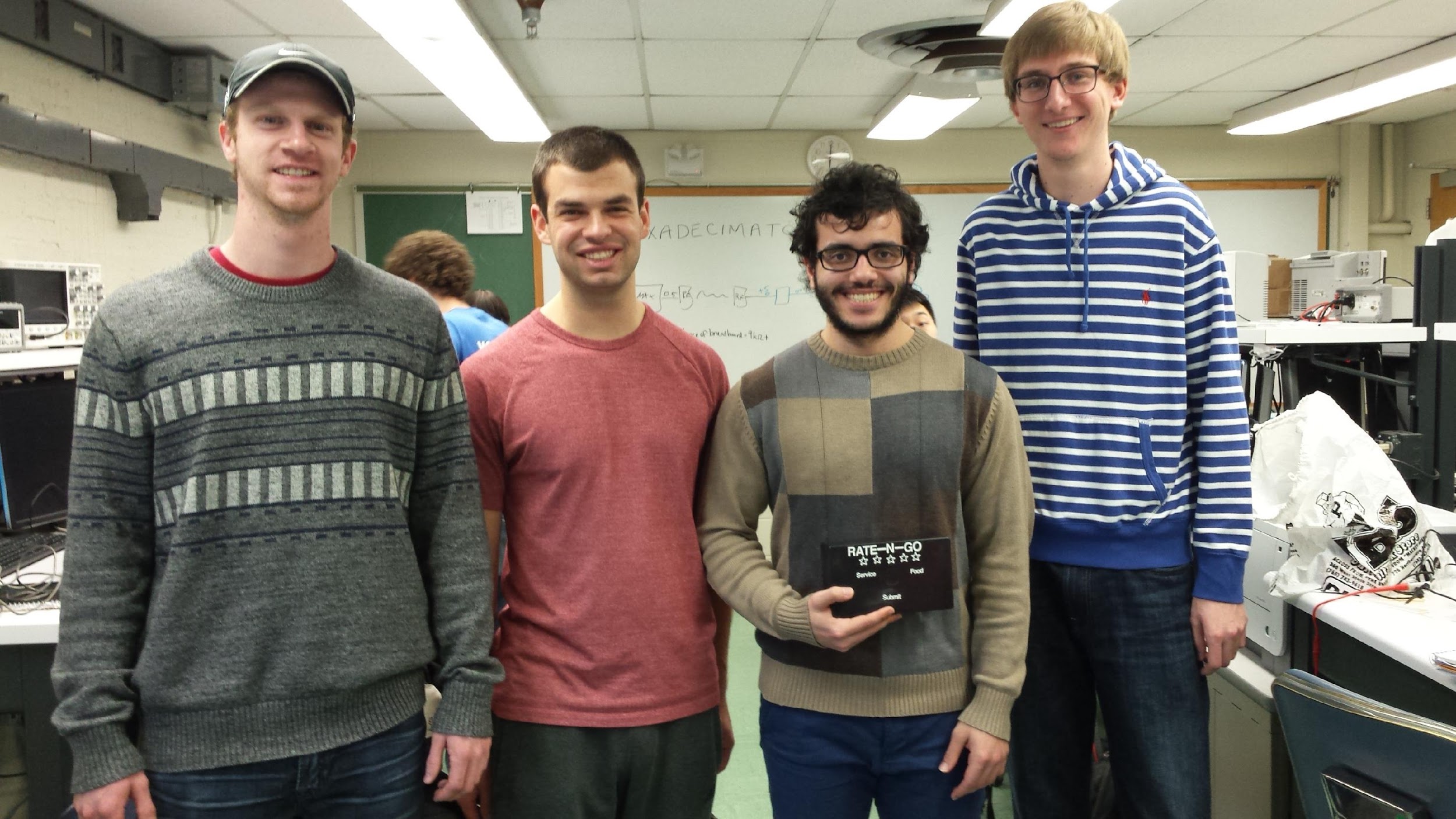
**<Rate ‘n Go>: Team 20**



|  |  |  |
| --- | --- | --- |
| *Team Members (left-to-right on picture, above)* | *Class No.* | *Lab Div* |
| Trevor Bonesteel | 1531-B | 4 |
| Dominic Celiano | 0617-C | 9 |
| Iaman Alkhalaf | 5528-A | 6 |
| John Mahony | 8160-M | 4youtu |

|  |  |
| --- | --- |
| *Report/Functionality Grading Criteria* | *Points* |
| Originality, creativity, level of project difficulty | 20 |
| Technical content, succinctness of report | 10 |
| Writing style, professionalism, references/citations | 10 |
| Project functionality demonstration | 20 |
| Overall quality/integration of finished product | 10 |
| Effective utilization of microcontroller resources | 10 |
| Significance of individual contributions\* | 20 |
| *Bonus Credit Opportunities* | *Bonus* |
| Early completion | 0.5% |
| PCB for interface logic | 2% |
| Poster (required for Design Showcase participation) | 1% |
| Demo video (required for Design Showcase participation) | 1% |
| Design Showcase participation (attendance required)\* | 1% |

##### \**scores assigned to individual team members may vary*

|  |  |
| --- | --- |
| *Grading Rubric for all Criteria (Including Bonus)* | *Multiplier* |
| *Excellent* – among the very best projects/reports completed this semester | 1.0 - 1.1 |
| *Good* – all requirements were amply satisfied | 0.8 - 0.9 |
| *Average* – some areas for improvement, but all basic requirements were satisfied | 0.6 - 0.7 |
| *Below average* – some basic requirements were not satisfied | 0.4 - 0.5 |
| *Poor* – very few of the project requirements were satisfied | 0.1 - 0.3 |

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| 1.0 Introduction | 1 |
| 2.0 Interface Design | 2 |
| 3.0 Peripheral Utilization | 4 |
| 4.0 Software Narrative | 5 |
| 5.0 Packaging Design | 7 |
| 6.0 Summary and Conclusions | 8 |
| 7.0 References | 10 |
| Appendix A: Individual Contributions and Activity Logs | 11 |
| Appendix B: Interface Schematic and PCB Layout Design | 20 |
| Appendix C: Software Flowcharts | 24 |
| Appendix D: Packaging Design | 27 |

1. **Introduction**

Our project is meant to provide immediate feedback to the staff of a restaurant from the consumer. This is accomplished by providing the consumer with a user-friendly interface to submit his or her rating of the restaurant quickly and easily (the “transmitter” side). This information from the transmitter side is then sent to the “receiver” side through RF communication, where an LCD screen displays the feedback. The information submitted by the user gives the restaurant feedback on three categories: Overall (1-5 stars), Food (red, yellow, or green), and Service (red, yellow, or green), where red is the worst and green is the best. These three categories provide ample information to the restaurant staff to see how they are doing overall, as well as what they need to work on in specific. Data can also be collected over many weeks to get an “overall” rating of the restaurant. This data can then be posted to a review site like Zagat or Google to have feedback from a larger sample size (as compared to the “convenience sample” of making someone log into a website to submit a rating).

In order to complete this project, each team member played a vital role. Dom was in charge of the overall organization of the team and the project, Trevor handled the hardware, and John and Ianman were in charge of the software.

One of the toughest parts of this project was planning everything out. Once we decided on an idea for the project, software and design considerations were made by the entire team, and the parts were ordered accordingly. Dom and Trevor designed a PCB for both the receiver and transmitter sides (two PCB’s total), and ordered them through Advanced Circuits. Once everything but the PCB’s had arrived, John and Ianman started to work on getting the software working using breadboards with wire. While they were doing this, Trevor worked on getting the boxes for the receiver and transmitter sides ready for the final product (drilling holes for the LED’s/pushbuttons, etc.). Throughout this process, Dom helped out with the wiring, coding, and hardware, and made sure everything was running smoothly. Once the PCB’s arrived, Trevor got the sockets soldered onto the boards and laid out all the wire needed on the transmitter and receiver sides. The entire team then worked together to fit everything into the boxes and get it all laid out properly. We then ran up the code John and Ianman had previously written and debugged the final product to ensure everything was working correctly. We then worked on writing the report, making the video, and ensuring all requirements of the project were met.

1. **Interface Design**

The interfaces utilized in this mini-project were pushbuttons, LEDs, potentiometers, piezo buzzers, and an LCD screen. On the transmitter side, there were 12 pushbuttons for user input and 12 LED’s. One LED corresponded to each pushbutton to display whether or not the pushbutton was pressed or not. The pushbuttons and LED’s were split up into 4 subsections: food rating (3 pushbuttons/LED’s), service rating (3), overall rating (5), and submit (1). On the transmit side, we decided to include a piezo buzzer, which would make a noise when the submit button was pressed. This was to notify the user of a successful submission. With the inclusion of this buzzer, we also decided to add a potentiometer to control the volume of the buzzer. We also had a reset pushbutton. We then created a schematic of the transmitter, including all of the interfaces mentioned above. The schematics for the transmitter can be seen in Appendix B.

In order to use the 9S12C32 Stamp Module to light 12 LED’s, we used a GAL 26v12 as a 12-bit shift register. This allowed us to use only two pins from the microcontroller to light the 12 LED’s. The other design consideration we had to make was how to sample the 12 pushbuttons, since there were not 12 inputs pins available for use on the microcontroller. We figured out that our configuration of 12 pushbuttons could basically be modeled as a 3x4 keypad, so we used the ECE 362 Lecture notes (CITATION) to figure out how to sample the 12 pushbuttons using only 7 pins GPIO on the microcontroller. The pushbutton sampling software is explained more in depth in the Software Narrative section, but it required us to have four 10kΩ pull-up resistors, which were included when the PCB was designed.

In order to use the SCI module, we needed to use a MAX3232 chip in our design because the stamp module changed the outputs of the microcontroller from CMOS levels (0 to +5V) to +9V to –9V. Because our RF receiver and transmitter need CMOS voltage levels to function, we needed a MAX chip on the transmitter side to change the voltages from -9V to +9V to CMOS levels. The schematics with and without the MAX chip can be seen in Appendix B. On the receiver side, it would have been ideal to have a MAX chip to change the voltages back to -9V to +9V, but it wasn’t 100% necessary. Since one of our MAX chips malfunctioned, we left it out.

On the receiver side, there were no LED’s or pushbutton inputs, but there was the LCD screen. This screen was used to output the ratings from the transmitter to the restaurant staff. In order to use this LCD screen, we used the SPI module from the 9S12C32 and an 8-bit shift register to send the data to the pins of the LCD screen. We also had pins from the microcontroller which were used to control the LCD (LCD Clock, Register Select, and LCD Read/Write). On the receive side, we used the same piezo buzzer and potentiometer system as the transmit side for creating a sound when data was received on the transmitter.

1. **Microcontroller Resource Utilization**

In our code, we utilized five of the microcontroller’s peripherals as well as the real time interrupt. The first peripheral we utilized was the pulse width modulation (PWM.) We used the PWM to create a variable voltage that we used to create sounds to confirm functionality on both the receiver side and the transfer side. We set MODR to 0x08 on the transmitter side in order to make sure that the buzzer on PT3 would be used as the PWM output. MODR was set to 0x01 on the transmitter in order to enable PT0 as the output for the PWM on the receiver. The same values were set for PWME and PWMPOL corresponding to each the transmitter and the receiver in order to make sure the proper channels were being used.

We then used a potentiometer to change the duty cycle in order to control the volume of the buzzers by changing the DC voltage. In order to change the duty cycle, we connected the potentiometer to an analog-to-digital (ATD) input. In order to initialize the ATD sequence, we set ATDCTL2 to 0x80 in order to enable the ATD. We set ATDCTL3 to 0x08 in order to initiate 1 conversion per sequence.  We did this in the timer interrupt service routine (TIM\_ISR.) We set TSCR1 to 0x80 in order to enable the power on bit. We set TSCR2 to 0x0C in order to reset TCNT when OC7 occurs and to set the prescale factor to 16.

We used the serial peripheral interface (SPI) to send data to the LCD screen similarly to how we have done in previous labs on the transceiver side. We used MOSI and SCK from PT6 and PT5 to provide a clock and data in to a 8-bit clock shift register to power the LCD on the screen on the receiver side. We used the same pins on the transmitter side on the transmitter side to control the sequence of our LED lights using a 10-bit clock shift register.

In order to get our transmitter and receiver to communicate with each other, we had to communicate using the serial communications interface (SCI.) The transmitter sent the data for the receiver from the TX pin on the SCI which the receiver acquired on the RX pin of the SCI. In order to do this, we set SCIBH to 0x07 and SCIBDL to 0x50 on both the transmitter and the receiver in order to set the baud rate to 2400 bit per second in order to stay within the accepted baud region on our RF module. In the SCI interrupt service routines on the transmitter and receiver we created a buffer that was used to store the characters being sent and received between the microcontrollers. Finally, the transmitter utilized the real time interrupt (RTI) service routine in order to sample the pushbuttons. We did this by setting the RTICTL register to 0x27.

1. **Software Narrative**

The software architecture for this project can be broken down into two main parts: the transmitter and the receiver. Each part is managed by a single microcontroller. The main task of the transmitter side is to capture the customer’s feedback and send that to the receiver side. Customers evaluate the service based on three criteria: Food, Service, and overall satisfaction. Food and Service can be rated on a scale of 3, and overall satisfaction can be rated on a scale of 5. Those ratings are obtained in an event-driven fashion through the use of pushbuttons. Each criteria has its own set of pushbuttons that span the whole scale of ratings. Once the customer enters all ratings, a submit pushbutton needs to be held in order to successfully send date to the receiver side.

Sampling the 12 pushbuttons is done using Real Time Interrupts (RTI). Sampling is accomplished through the use of seven pins. Three of the pins are programmed as outputs, and alternate so that only one of the three pins is low every time an RTI interrupt occurs. The other four pins are programmed as inputs. Each pushbutton is represented by a combination of an output port being low, and an input port switching state from high to low, leading to a total of 12 possible pushbutton combinations.

Each pushbutton has a corresponding LED. LEDs are being clocked manually in the program by an output port pin and a data pin. The clock and data pin go into a 12-bit shift register. Each time a pushbutton is pressed, an LED sequence is determined and then clocked through the use of a dedicated clocking function that receives the required LED sequence and clocks appropriately. After the customer holds the submit button, a ‘ready’ flag set to indicate the transmitter is done gathering data. After that, a fixed start byte and three other bytes based upon ratings are sent to a buffered character output routine BCO. BCO determines if the buffer is full or empty, places the character in the buffer, and enables the transmitter interrupt to see if data are ready to be sent to the receiving microcontroller.

The receiver is responsible for capturing ratings entered by the customer from the transmitter side and displaying a summary of those ratings on an LCD screen. A buzzer is activated based upon a successful date receive. The volume of the sound is adjustable through adjusting the PWM duty cycle using a potentiometer connected to ATD input 0. The receiver reads the four bytes received through using a buffered character input routine in a manner analogous to the transmitter side. The receiver maps the four received bytes (including the start byte) to the appropriate ratings. This is done through detecting which received byte of the four is a start byte and that constitute what the remaining three bytes refer to. The Receiver then calls a display function that uses the SPI module to send data to the LCD.

1. **Packaging Design**

In order to package our product, we had to consider both the receiver and transmitter sides. We drew up an initial design on how the transmitter side would work (seen Appendix D-1), and worked as a team to figure out how we were going to design and implement our theoretical idea into an actual product. We decided to order two project boxes online (see Appendix D-2) which would hold the components for the transmitter and receiver, as well as the two PCB’s and all of the wiring.

For the transmitter side, we had to lay out our packaging so we could view the 12 LED’s as well as receive input from the 12 pushbuttons. We decided we would put all these buttons and LED’s on the front of the box, with the wires sticking into the interior of the project box. We decided to do the same thing for wires of the potentiometer and reset button, but these would instead be on the side of the box. Pictures of this layout can be seen in Appendix D. Once the wires had been sticking through to the inside of the box, we needed to solder all of the wires together and route them to the PCB inside the transmitter. Then to power the microcontroller, we used a barrel jack to take an input from a 5V DC voltage from a wall jack. That power was then soldered onto the board. All of this wiring was very time consuming and looked ugly, but since it was inside the box, it didn’t matter much. The outside of the box looked pretty and that was all that mattered.

For the receiver side, we didn’t have nearly as much to worry about. We were using the same project box, but all we needed to put on the project box was the LCD and the normal reset button and potentiometer (which were also on the transmitter side). Once again, all the wires were put into the middle of the box, the LCD plugged right into the PCB, and the wires from the potentiometer and reset button were soldered together and tied to the female headers on the PCB. Once again, this looked ugly, but it didn’t matter because it was all inside the box. In order to power the receiver, we didn’t bother using a barrel jack because the receiver end didn’t need to look that pretty, since it was only being viewed by the restaurant staff. We just tied a 5V power supply straight to the PCB and soldered it on. All in all, our project looked pretty good and the PCB’s made everything a lot easier to package together. The project boxes really came in clutch, and made our project look really good to both the consumer and receiver.

1. **Summary and Conclusions**

Doing a design project from start to finish with a deadline to meet taught us lessons that will be invaluable to us as future engineers. The first is that no matter how well we plan, we will almost always run into unexpected issues. Therefore, it is necessary to leave ample time so we can still meet our deadline. One issue we ran into was getting the SCI to communicate between the two microcontrollers to work properly. Our code did not work the first time we tested it, and so we learned how valuable it is to use programs like Tera Term and simple inchar/outchar functions to debug, rather than seeing how values change in the CodeWarrior debugger. Yet because we started our software work well before the project was due, we were able to solve all the issues we ran into, including replacing microcontrollers that weren’t working as expected.

Another lesson we learned was how to make the most of out the input and output pins on our microcontroller. When we first thought of how to sample 12 pushbuttons without having that many input pins on the 9S12, we were clueless. But after started to research and talk to others we started learning how to do it. The same thing happened for lighting up 12 LED’s. Going from asking a question we didn’t know the answer to to researching, designing, building, and debugging has been an extremely valuable process and one of the most important things we have learned from doing this project. Another example of a question we asked was how to do wireless communication. We have grown up around wireless communication for our entire lives, but having to use it ourselves in our Mini-Project really allowed us to learn how it worked. Before we could even think about our design, we had to explore the different options of how to communicate wirelessly, and then decide which methods were simply too expensive (i.e. Xbee), and which ones would work (RF). Once we chose to use RF to communicate, we learned about how the RF devices and protocol worked, and how to interface them with the 9S12.

If we had more time to improve our design, a lot of the improvements we would make would be with the hardware. Once our hardware arrived, we had to spend hours soldering everything together, as well as laying out the transmitter and receiver and gluing together the different components. If we had more experience with designing products and working with electrical components (like we have gained now), we could have saved hours of time by designing a better PCB or using different wire configurations.

As far as our product as a whole goes, some improvements we can make to it would be to make a friendlier user interface and a cleaner overall layout. Another feature we can add in the future is to connect our transmitter and receiver to a web server and write a script to update websites like Google Ratings and Zagat to give live restaurant ratings online. This would add complexity to our project, but the basis we have laid with our project would make adding a feature like that quite simple. Having that functionality would revolutionize the Internet Rating system and would change the internet (as well as restaurants) for the better.

**References**

[1]J. Gray, 'KLP/KLP Module Walkthrough', SparkFun Electronics, 2015. [Online]. Available: https://www.sparkfun.com/datasheets/RF/KLP\_Walkthrough.pdf. [Accessed: 11- Dec- 2015].

[2]SP3222EB/3232EB, 1st ed. Sipex, 2015.

[3]H. Huang, The HCS12 / 9S12: An Introduction to Software and Hardware Interfacing, 2nd ed. Cenage Learning, 2009.

[4]Wireless Hi Sensitivity Receiver Module (RF ASK), 1st ed. WENSHING®©.

[5]Wireless Hi Power Transmitter Module (RF ASK), 1st ed. WENSHING®©, 2015.

**Appendix A:**

**Individual Contributions**

**and**

**Activity Logs**

**Activity Log for:** Dominic Celiano **Role:** TDP Leader

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Activity*** | ***Date*** | ***Start Time*** | ***End Time*** | ***Time Spent*** |
| Initial Team Meeting | 15 Nov | 1900 | 2000 | 1 Hour |
| Software Considerations Meeting | 21 Nov | 1900 | 2100 | 2 Hours |
| Schematic Design/PCB Layout/Design | 22 Nov | 1700 | 2330 | 6.5 Hours |
| Final PCB Design/Layout | 23 Nov | 1700 | 2200 | 5 Hours |
| Software/Building Circuit on Breadboard | 5 Dec | 0900 | 1900 | 10 Hours |
| Helping debug software | 8 Dec | 1900 | 2000 | 1 Hour |
| Soldering and Hardware Building | 9 Dec | 1800 | 2100 | 3 Hours |
| Poster and Report Work | 8-9 Dec | Misc. | Misc. | 4 Hours |
| Poster/Report/Building/Soldering | 9 Dec | 1130 | 1730 | 6 Hours |
| Report Writing/Helping with Hardware | 10 Dec | 2000 | 2230 | 2.5 Hours |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Written Summary of Technical Contributions:** Dominic Celiano

One of the hardest parts of this project was getting everything going from the beginning: the deadline was far away and we had other ECE 362 labs to worry about. But I took some initiative on the design of the project and made sure everyone was on board with understanding why it was so important to get our design done early so we could get our schematics done and our parts and PCB’s ordered early. I helped our hardware guy, Trevor, get the Transmitter PCB design laid out in Eagle before Thanksgiving, while I made the Receiver schematic and PCB design. I also organized a couple team meetings/set up a GroupMe before Thanksgiving to make sure everyone was on the same page.

Once our RF modules arrived, we were able to start wiring everything up on our breadboards and test our different components. Trevor wired up the Transmitter side, while I wired up the receiver side and we waited for the software guys to arrive. Once they arrived, Trevor and I discussed some hardware considerations, and then I helped John and Iaman get some of their code working and helped them understand the schematics Trevor and I had made while laying out the PCB’s. One of the biggest problems John and I ran into as soon as we started debugging our components was how to program the GAL26V12. We couldn’t use the normal Dataman in the lab, so I figured out we can use the Dataman in the ECE 477 Senior Design Lab. We moved forward from there. After multiple hours of staring at ABEL code to get the GAL to work, we eventually got it to work and moved forward on how to sample all the pushbuttons inputs and tie them to the LED outputs. I pointed John towards some example code I found online, and we got it working via the debug window.

After that initial software session, John and Iaman took charge of getting the SCI to work, while Trevor and I planned out some more hardware. I made some decisions on how to get our project done on time, like starting to write the report, make the poster, and ensure all our parts were working. We met multiple times throughout the week (see activity log) to work on everything and make sure everything was getting done in a timely fashion. Towards the end of the week, I spent a lot of time in lab getting our report/poster done as well as helping with whatever hardware and software I could.**Activity Log for:** Iaman Alhkalaf **Role:** Software Leader

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Activity*** | ***Date*** | ***Start Time*** | ***End Time*** | ***Time Spent*** |
| Initial Team Meeting | 15 Nov | 1900 | 2000 | 1 Hour |
| Software Considerations Meeting | 21 Nov | 1900 | 2100 | 2 Hours |
| Wiring the receiver side software, debugging breadboard circuit | 5 Dec | 1200 | 2200 | 10 Hours |
| Collaborating with Peripheral leader to configure transmitter/receiver communication | 6 Dec | 1200 | 2200 | 10 Hours |
| Consulting TA’s | 7 Dec | 1900 | 2200 | 3 Hours |
| Collaborating with Peripheral leader to configure transmitter/receiver communication | 8 Dec | 1200 | 2200 | 10 Hours |
| Exploring the RF module, software changed | 9 Dec | 1630 | 1830 | 2 Hours |
| Verifying software after installing PCB’s | 10 Dec | 1230 | 1430 | 2 Hours |
| Writing Appendix C and Section 4.0 of Report | Misc. | Misc. | Misc. | 3 Hours |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Written Summary of Technical Contributions:** Iaman Alhkalaf

Mainly responsible for writing the software for the receiver side of the project. Spent a significant amount of time configuring and debugging the transmitter/receiver interaction through SCI. Verified project’s functionality using wired communication. Analyzed circuit’s behaviour after installing the RF module and adjusted the software accordingly. **Activity Log for:** Trevor Bonesteel **Role:** Interface Leader

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Activity*** | ***Date*** | ***Start Time*** | ***End Time*** | ***Time Spent*** |
| Initial Team Meeting | 15 Nov | 1900 | 2000 | 1 Hour |
| Schematic Design/PCB Layout/Design | 22 Nov | 1700 | 2330 | 6.5 Hours |
| Final PCB Design/Layout | 23 Nov | 1700 | 2200 | 5 Hours |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Written Summary of Technical Contributions:** Trevor Bonesteel

*Provide a concise but sufficiently detailed description of your technical contributions to the project.*

*Length should be about one page.***Activity Log for:** John Mahony **Role:** Peripheral Leader

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Activity*** | ***Date*** | ***Start Time*** | ***End Time*** | ***Time Spent*** |
| Initial Team Meeting | 11/15 | 3:30 | 4:30 | 1 |
| Software Considerations Discussion | 11/21 | 4:00 | 7 | 3 |
| Programming the Receiver | 12/5 | 12:00 PM | 12:00 AM | 12 |
| Programming/Debugging | 12/6 | 2:00 PM | 12:00 AM | 10 |
| Programming/Debugging | 12/8 | 4:00 PM | 10:00 PM | 6 |
| Software/Debugging | 12/9 | 2:00 PM | 4:00 PM | 2 |
| Report/Debugging/Hardware | 12/10 | 11:30 | 5:30 | 6 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Written Summary of Technical Contributions:** John Mahony

During the early stages of the project, I was involved in the initial discussions on how we would potentially have to wire up our microcontrollers in order to ensure a functional. I suggested potential hardware that we could use for our final project such as the GAL26V12 we ended up using to light our LED sequence.

Once we got out of the planning phase of our project, I was primarily involved with the software aspect of the project. I primarily programmed the transmitter side of our project. I figured out how to write the code to cycle through the keypad like sequence we needed in order to make the amount of push buttons we had work. I wrote the code to assign a value to all the necessary being sent to the receiver. I also made sure that values were only being sent after the submit button was pressed. I also programmed the GAL26V12 in order to be a 10-bit shift register necessary for displaying our led sequence. I also enabled the volume and sound coming out of our speaker after the submit button is pressed.

The biggest challenge that I faced was dealing with the SCI to transmit data to the receiver. Ayman and I had much trouble with this at first and spend a huge chunk of time trying to debug the issues we were having. We eventually figured it out after learning about certain inadequacies in the debugger. Once our team tried to hook up to the wireless RF module on our device, we discovered that the MAX chip on our receiver side was not functional. Once this happened Ayman and I had to redo some of our code to account for the inaccuracies in the received data.

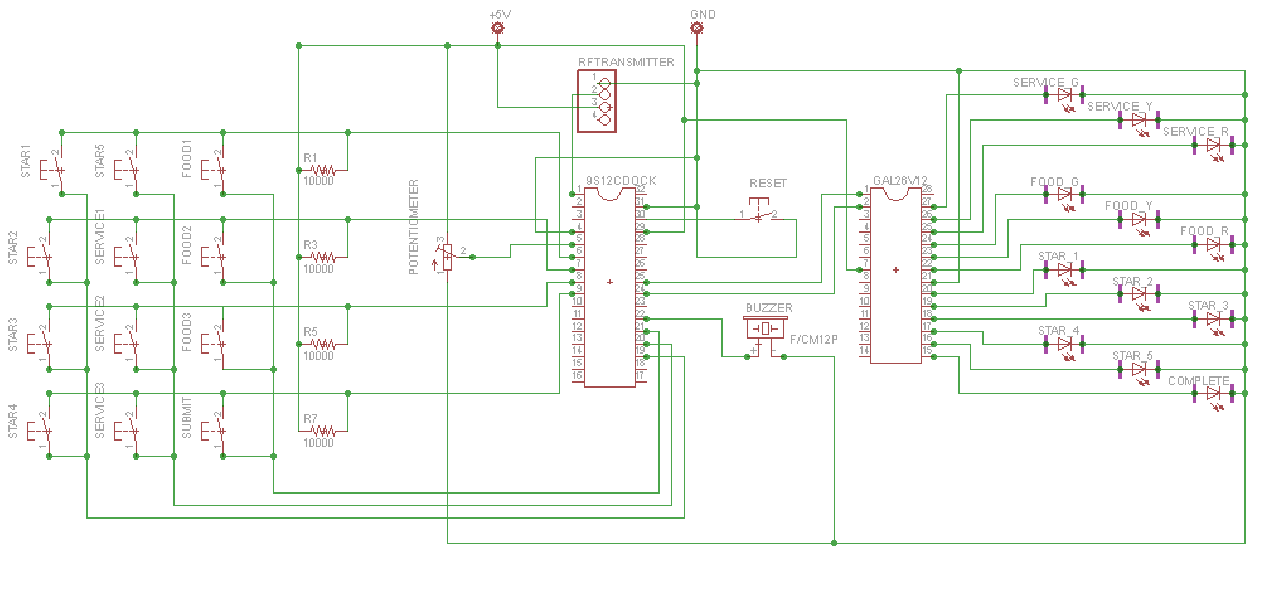
Once all the software was written I assisted Trevor and Dom with their hardware installation. This included some soldering as well as installing the final PCB’s in their proper enclosures. I also wrote the Microcontroller Resource Utilization of the final report.

**Appendix B:**

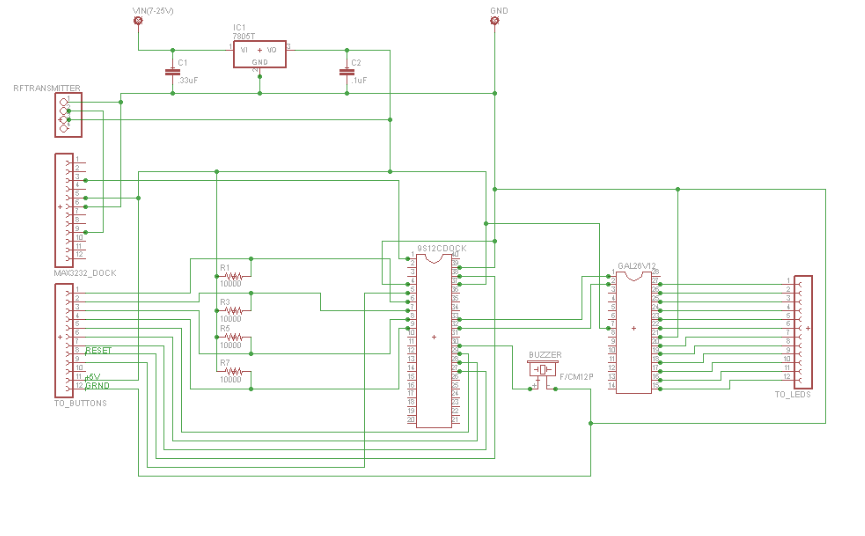
**Interface Schematic**

**and**

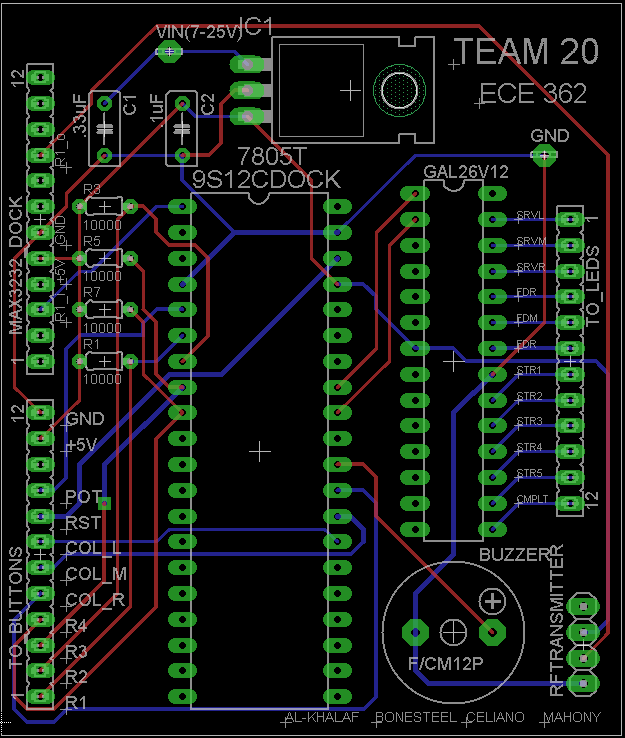
**PCB Layout Design**

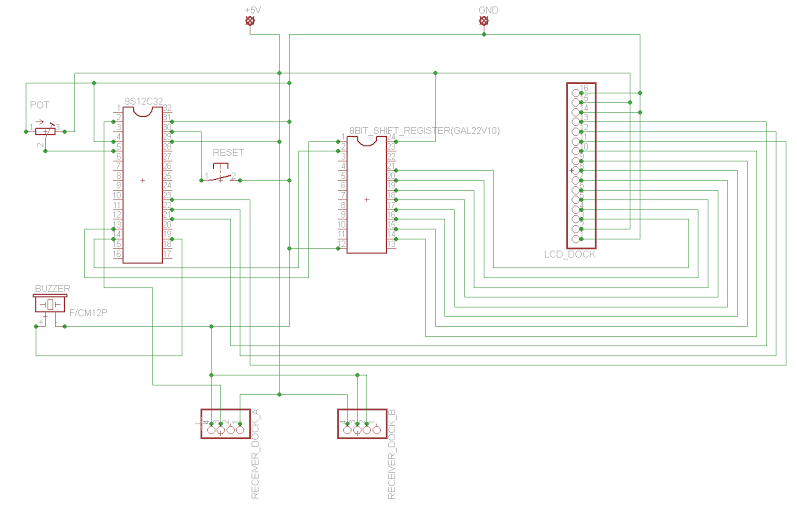
**B – 1:** Transmitter Schematic showing LED/Pushbutton Array (without MAX3232 Chip) – made by Trevor Bonesteel

**B – 2:** Transmitter Schematic showing header connections and MAX3232 connections – made by Trevor Bonesteel

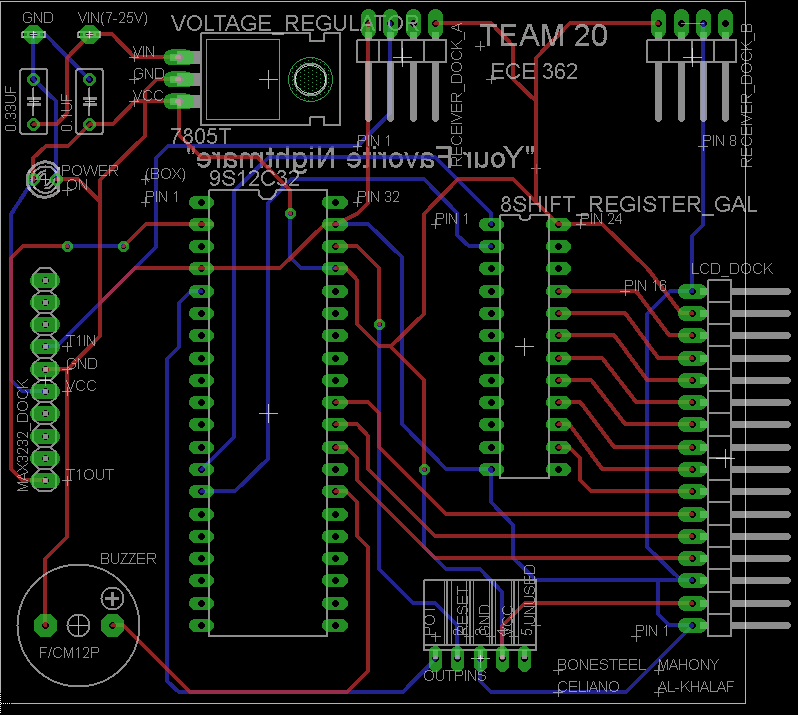


**B – 3:** Transmitter PCB Layout produced by Eagle by Trevor Bonesteel



**B – 4:** Receiver Schematic without MAX3232 connections (since the MAX wasn’t actually necessary) – produced by Dominic Celiano

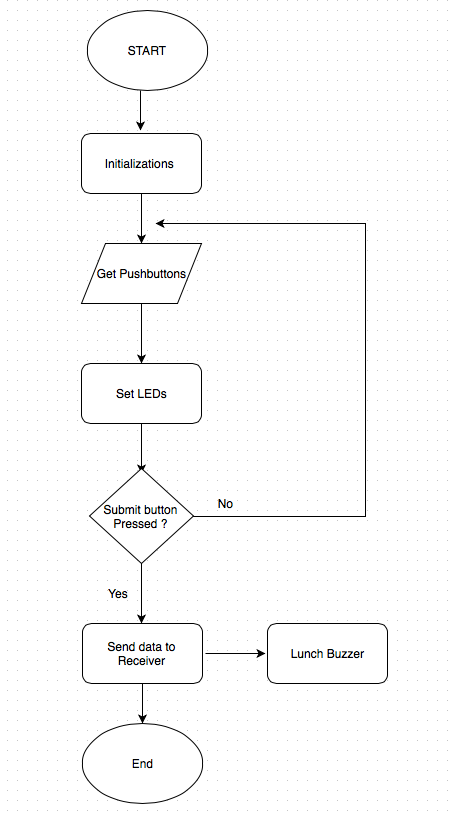
**B – 5:** Receiver PCB Layout, produced in Eagle by Dominic Celiano



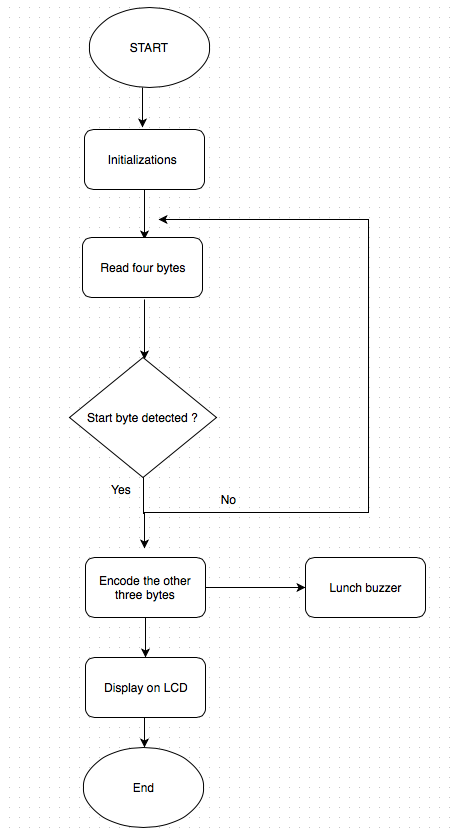
**Appendix C:**

**Software Flowcharts**

**C - 1**: The Transmitter: flowchart made by Iaman Alkhalaf

**

**C - 2**: The Receiver: flowchart made by Iaman Alkhalaf

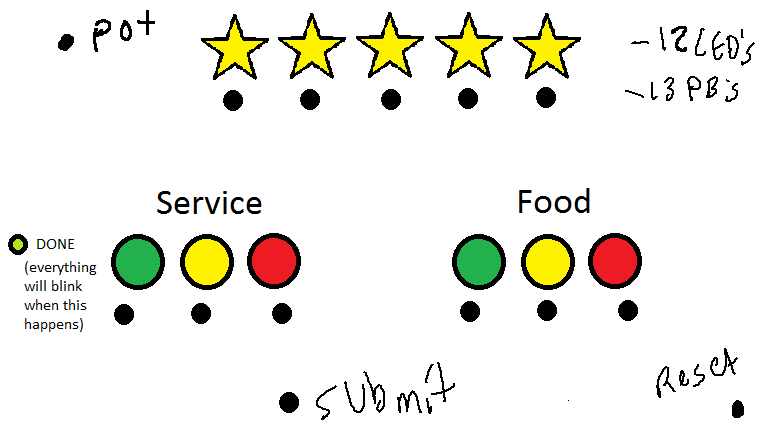
****

**Appendix D:**

**Packaging Design**

*All documentation/pictures in this Appendix produced/taken by Dominic Celiano*

**D – 1:** Transmitter layout used to plan



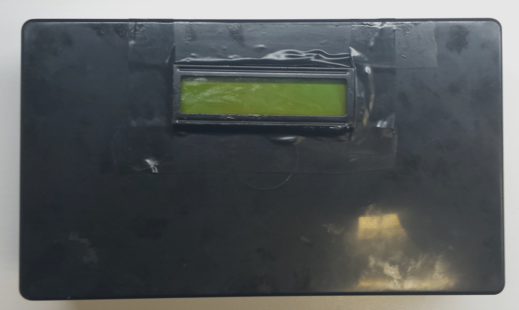
**D – 2:** The bare box ordered for use on the Transmit and Receive sides



**D – 3:** The box used for the Transmit side



**D – 4:** The box used for the Receive side



**D – 5:** The Potentiometer and Reset Button shown on the box

