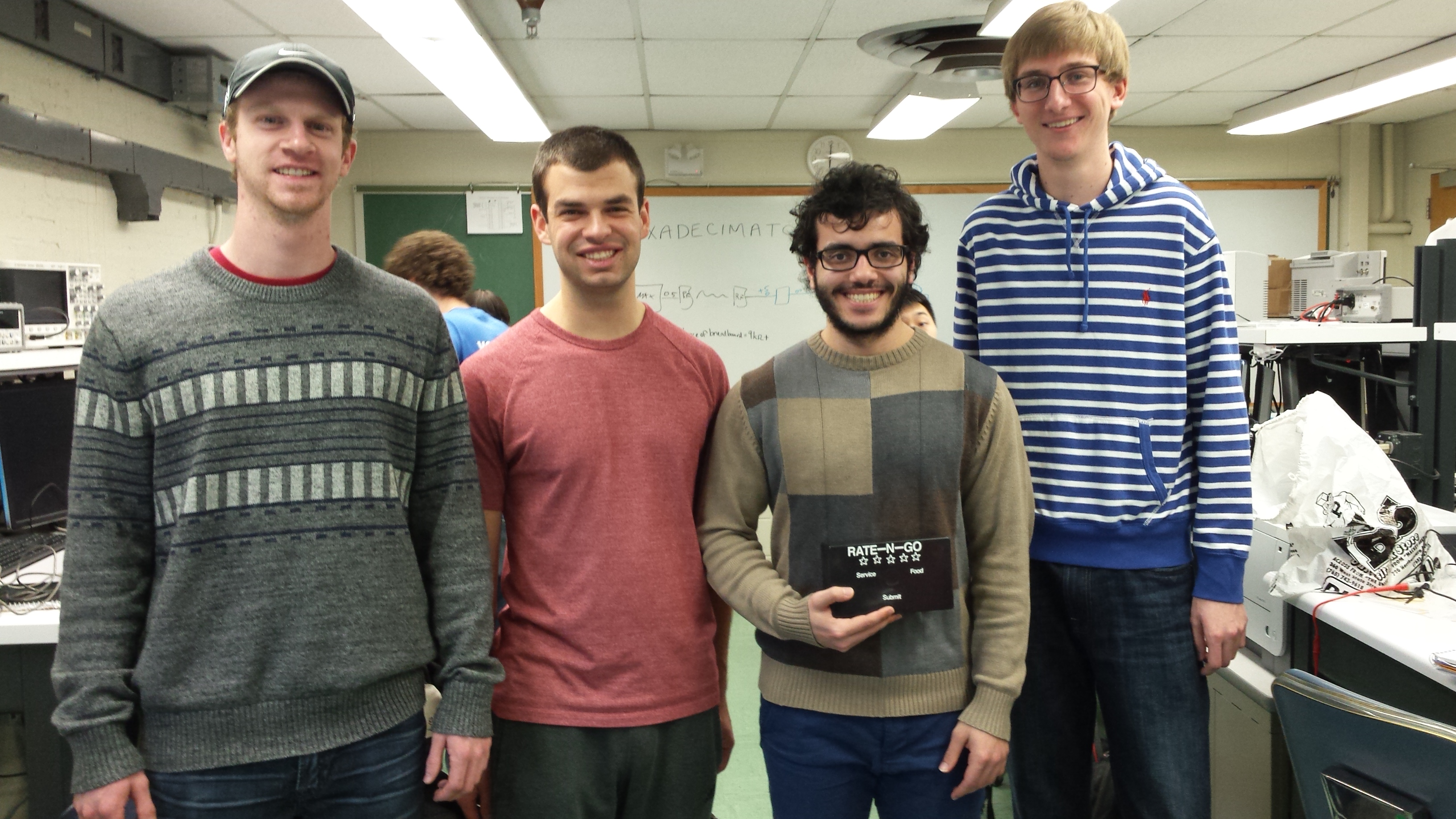
**<Rate ‘n Go>: Team 20**



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

|  |  |
| --- | --- |
| *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 | 3 |
| 4.0 Software Narrative | 4 |
| 5.0 Packaging Design | 5 |
| 6.0 Summary and Conclusions | 6 |
| 7.0 References | 7 |
| Appendix A: Individual Contributions and Activity Logs | 8 |
| Appendix B: Interface Schematic and PCB Layout Design | 17 |
| Appendix C: Software Flowcharts | 19 |
| Appendix D: Packaging Design | 21 |

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 the real time interrupt. The first peripheral we utilized was the pulse width modulation (PWM.) We used the PWM to to create a variable voltage that we used to create an 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 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 reciever acquired on the 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 reciever 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 reciever we created a buffer that was used to store the characters being sent and recieved 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**

*Describe what you learned from completing the project and what you might do to improve your design if you had more time.*

*Length should be about one page.*

Here is a summary of the most significant challenges we had and the lessons we learned through overcoming them:

* Challenge 1: interfacing 12 pushbuttons and LEDs
  + Lesson learned: “interface people, write something here”
* Challenge 2: Using SCI to communicate between two microcontrollers
  + Lesson learned: when working with multiple microcontrollers, using tera term and simple inchar/outchar messages is more efficient than stepping through the code or observing values as they update in the debugger.
* Challenge 3: High costs of reliable communication methods (i.e Xbee)
  + Solution: replace the with RF transmitters, much cheaper but requires additional software adaptation
  + Lesson learned: cost can be minimized if there is a will to spend more time adjusting hardware/software to cheaper parts.
* Challenge 4: How to lay everything out
* Challenge 5: Getting the GAL26V12 to work/finding something to program it
* Challenge 6: Soldering wires together – maybe we should have just gotten a huge PCB
* We had a broken max chip – it turned out to not actually be necessary, though
* Lessons/Improvements to make:
* Get a bigger PCB and put everything on there
* Nicer display/user interface
* A “first time?” button
* Something to connect to the internet to update Google Ratings
* Nothing is easy as it seems in theory

1. **References**

*List any references (e.g., data sheets, application notes, web sites) used in formulating your solutions.* ***Be sure to cite these references in your report.***

*NOTE: Use IEEE format.*

**Appendix A:**

**Individual Contributions**

**and**

**Activity Logs**

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Activity*** | ***Date*** | ***Start Time*** | ***End Time*** | ***Time Spent*** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

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

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

*Length should be about one page.***Activity Log for:** Iaman Alhkalaf **Role:** Software Leader

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Activity*** | ***Date*** | ***Start Time*** | ***End Time*** | ***Time Spent*** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

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

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

*Length should be about one page.***Activity Log for:** Trevor Bonesteel **Role:** Interface Leader

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Activity*** | ***Date*** | ***Start Time*** | ***End Time*** | ***Time Spent*** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**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*** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

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

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

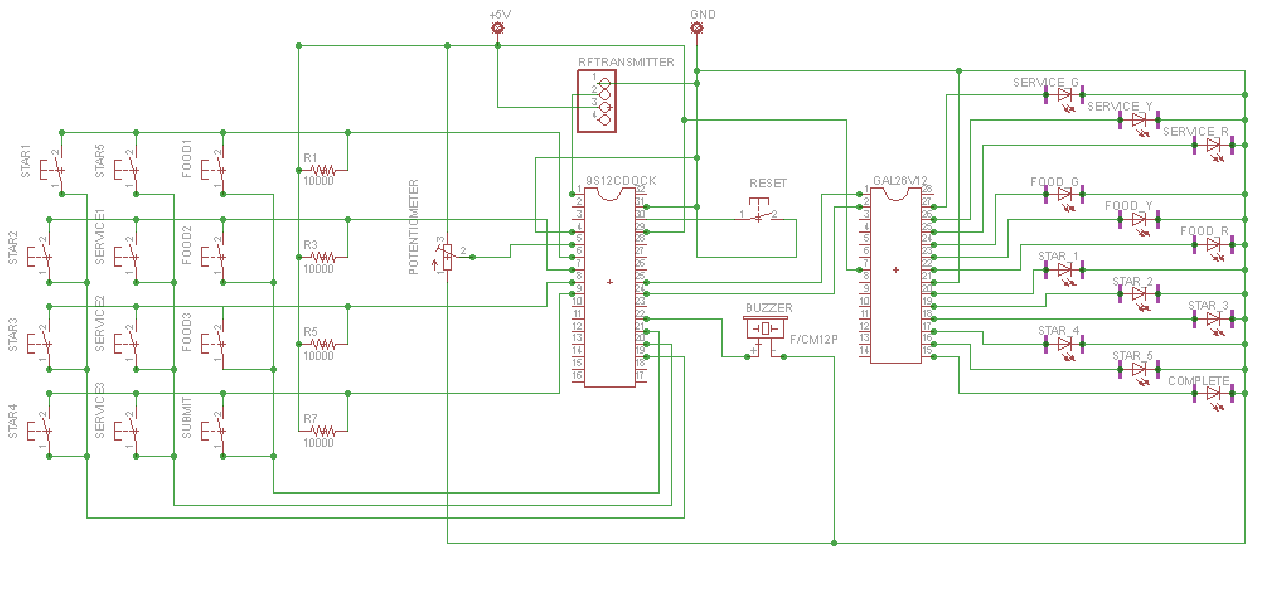
*Length should be about one page.*

**Appendix B:**

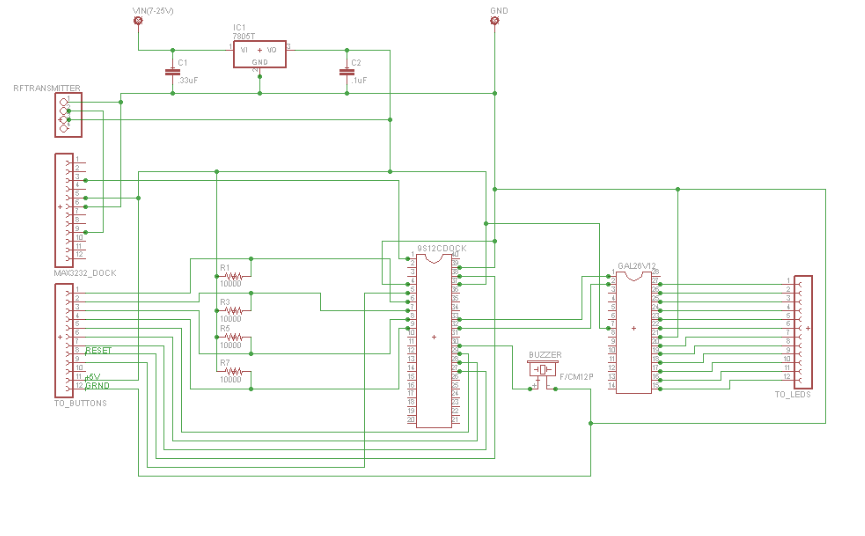
**Interface Schematic**

**and**

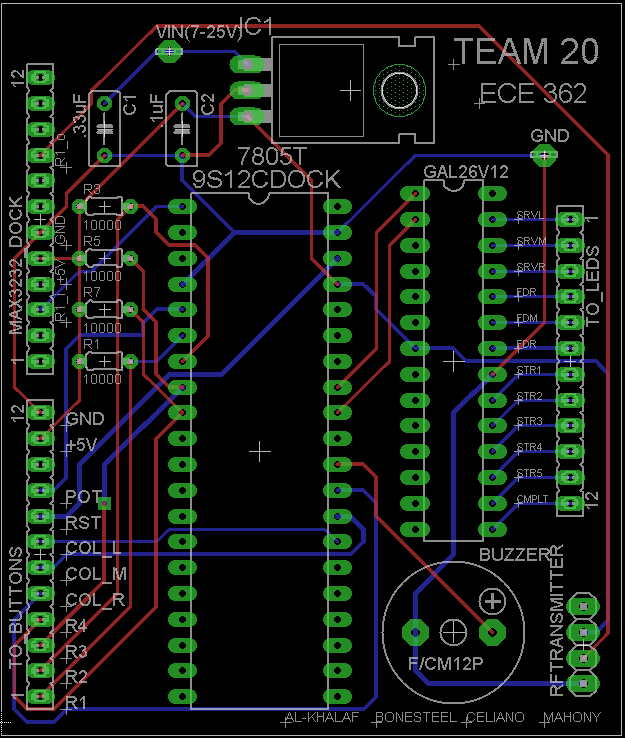
**PCB Layout Design**

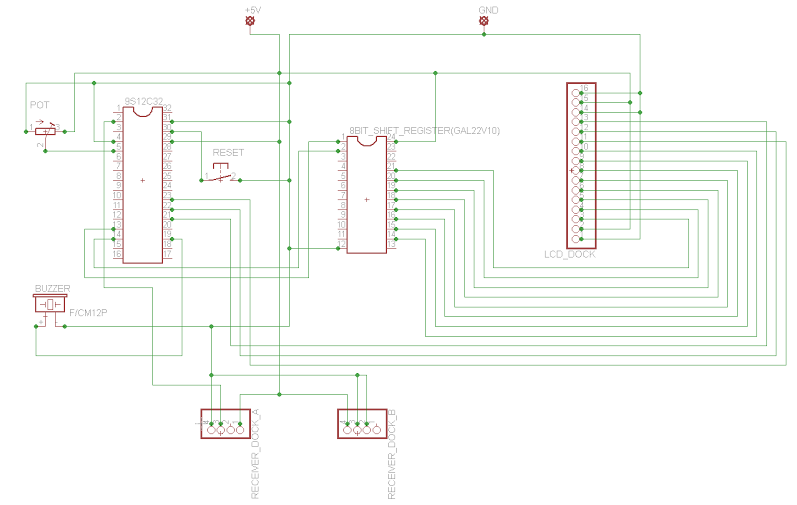
**B – 1:** Transmitter Schematic showing LED/Pushbutton Array (without MAX3232 Chip) – made by Trevor Bonesteel

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

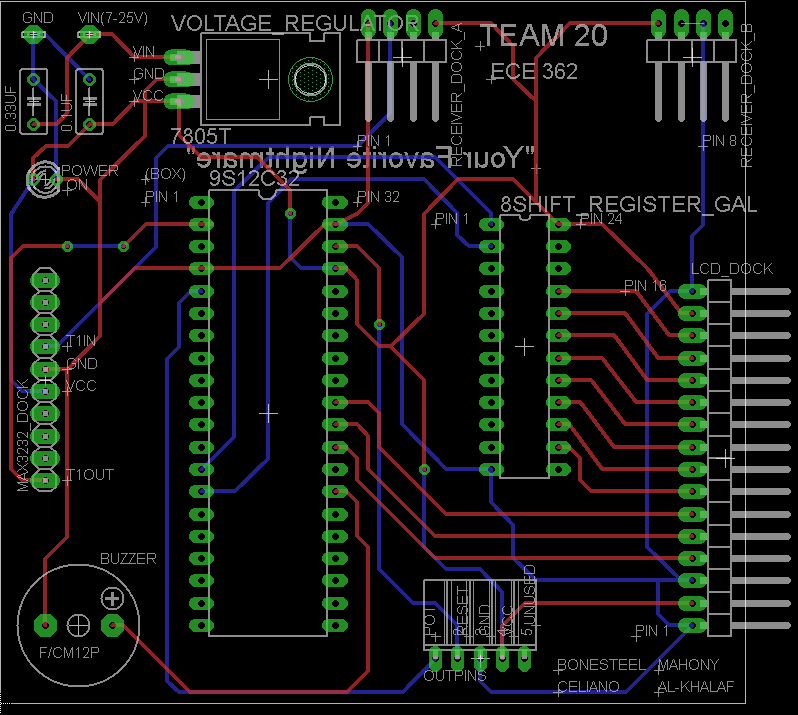


**B – 3:** Transmitter PCB Layout produced by Eagle by Trevor Bonesteel



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

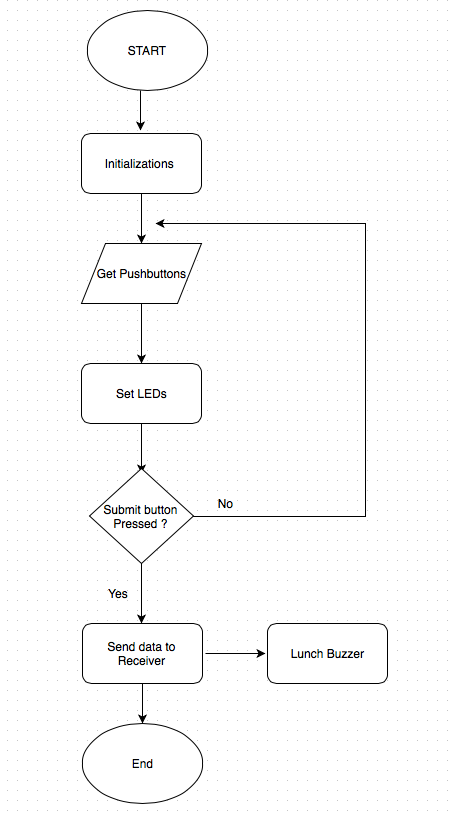
**B – 5:** Receiver PCB Layout, produced in Eagle by Dominic Celiano



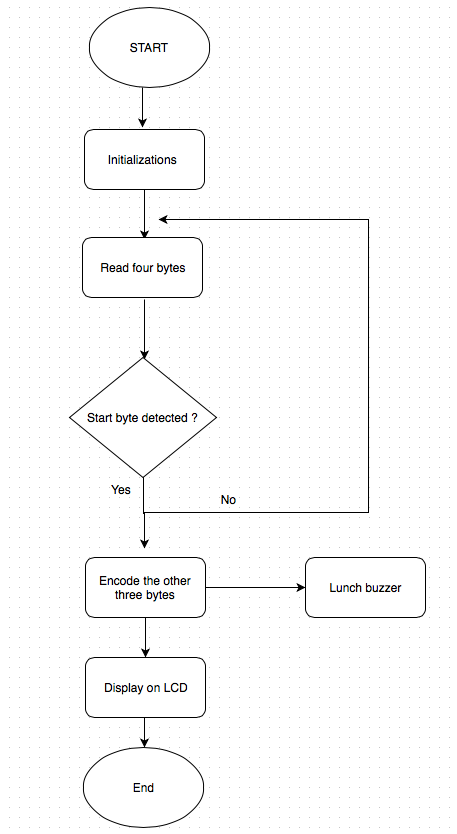
**Appendix C:**

**Software Flowcharts**

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

**

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

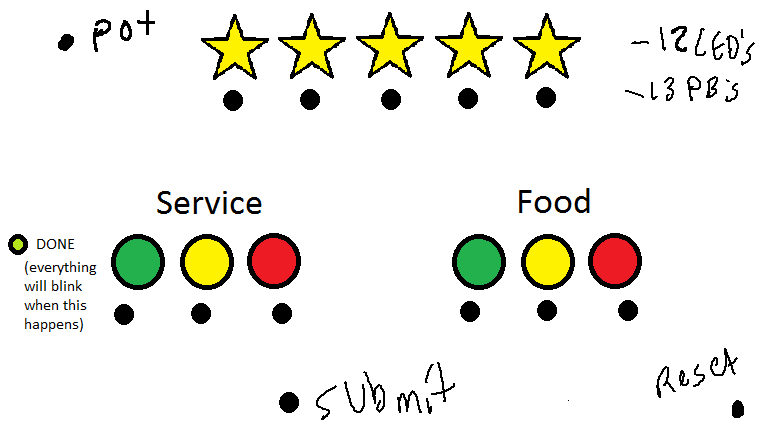
****

**Appendix D:**

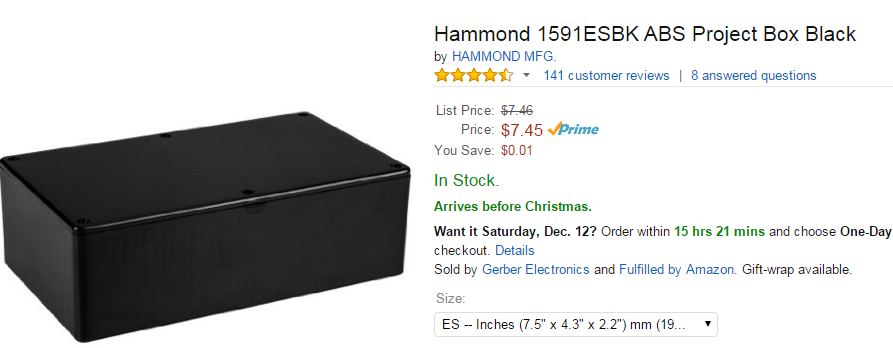
**Packaging Design**

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

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



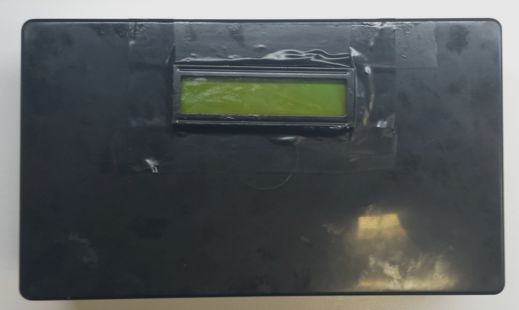
**D – 2:** The bare box ordered for use on the Transmit and Receive sides



**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

