Senior Design 2017

MY SHELTER

Elizabeth Dominguez
Miguel Fernandez

Project Abstract

MYshelter is an animal shelter management system that keeps track of the pets that are surrendered and adopted. Each animal is given an RF tag with a unique ID, like the one shown in Figure 1. The ID is scanned once when the animal is surrendered to the shelter, and a second time once the animal is adopted. This information will be sent to a database that keeps track of the animals and other relevant information. An example of what may appear on the database is shown in Table 1.

FIGURE 1 RF TAG GIVEN TO EACH PET

TABLE 1 INFORMATION STORED IN DATABASE

ID	TYPE	BREED	STATUS
12345	DOG	HUSKY	SURRENDERED
23456	CAT	TABBY	ADOPTED

The information on this database will then be used on an adoption website Figure 2. The website will showcase pictures of the available pets. If an animal is adopted, its picture will be blurred out on the website once the page is refreshed. The website also has an administration page that allows users to change the IDs of each dog without having to scan the tag.

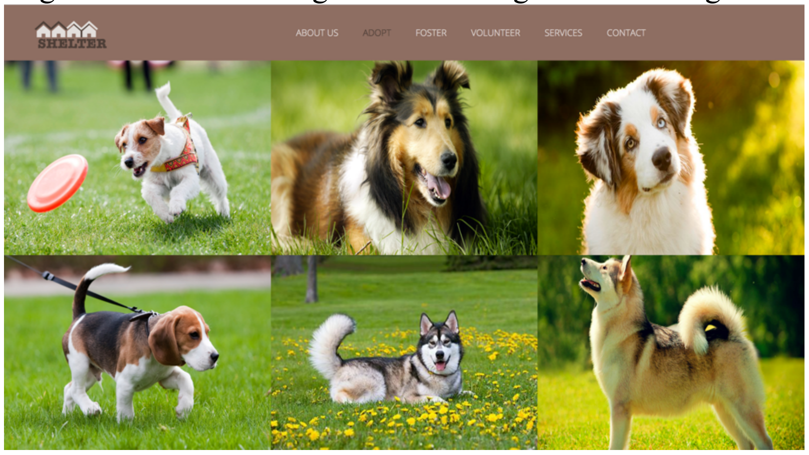


FIGURE 2 THE MYSHELTER WEBSITE

Project Features

This design requires RFID tags, a receiver circuit, and a Wi-Fi module. The receiver will use a 125kHz carrier signal to power the RFID tag via an antenna. An AM signal demodulator will retrieve the stored information on the RFID tag in digital form. The receiver must obtain information accurately with minimal interference. Amplifiers will be used in the receiver circuit to amplify the analog signals for better data interpretation. The microprocessor will then retrieve the signal information and decode it.

Our design will be split into two systems: the mainboard with the Wi-Fi module, and a handheld RF module. Both systems use an LCD and XBEEs to communicated via UART. The receiver module is also hooked up to an AB amplifier and an 8 ohm speaker that beeps whenever a card is successfully read. Our system is powered by two 6V wall plugs and on/off switches.

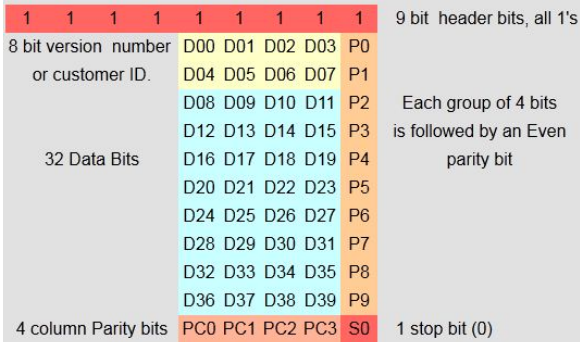
Concept & Technology Selection

Our design will include the following components:

- 2 Atmega644PA
- RF Antenna
- RFID cards
- Receiver
- 2 LCDs
- 2 XBEEs
- Wi-Fi module
- AB amplifier and 8 ohm speaker
- Power Connectors

RF Reader

Our RF reader reads 125 kHz tags and uses the EM4100 protocol to produce level transitions in the middle of a bit period. A low to high transition signifies a logic of 1, and a high to low transition signifies a 0. When the tag approaches the electromagnetic field of the antenna, it sends out a stream of bits shown in Figure 3. The first 9 bits are 1s, used to indicate the beginning of a read. They are followed by 8 customer ID numbers, each with their own parity bits. Then 32 data bits and their parities are sent as the card ID. Finally, 4 column parity bits and 1 stop bit are transmitted for a total of 64 bits.



1 1 1 1	1	1	1	1	1	9 bit header bits, all 1's
8 bit version number	D00	D01	D02	D03	P0	
or customer ID.	D04	D05	D06	D07	P1	
	D08	D09	D10	D11	P2	Each group of 4 bits
	D12	D13	D14	D15	P3	is followed by an Even
32 Data Bits	D16	D17	D18	D19	P4	parity bit
	D20	D21	D22	D23	P5	
	D24	D25	D26	D27	P6	
	D28	D29	D30	D31	P7	
	D32	D33	D34	D35	P8	
	D36	D37	D38	D39	P9	
4 column Parity bits	PC0	PC1	PC2	PC3	S0	1 stop bit (0)

FIGURE 3 THE TRANSMITTED BIT STREAM

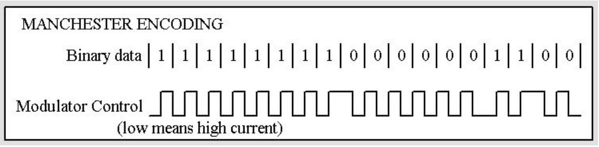


FIGURE 4 MANCHESTER ENCODING

In order to combat the noise in our receiver circuit, we attached bypass capacitors to power lines in addition to a Schmitt trigger. A Schmitt trigger is a comparator with hysteresis that converts analog to digital. Its output changes only when the input changes past a second threshold.
A 125kHz square wave is generated via counter 2 on PIND7 and fed into the input of the receiver. The output of the receiver is then connected to an input pin of the microcontroller. Using a 16-bit, timer, we store the logic value ever 500 us into a large buffer. Then a counter counts each set of 9 1s—whenever the input pin reads 0, it resets the count to 0 to only account

for consecutive 1s. Another array then stores where each consecutive 9 1s are in the buffer and increments this value to begin at the first value of the 50 card ID values and parity bits.
Once the buffer is full, the data is decoded. Each bit is shifted 4 times to create a 4-bit integer. The parity and column parity bits are saved. We then calculate what the column and row parity bits should be in a separate loop. If the scanned and calculated parity bits do not match, we do not print the value. Only when we have a perfect read to we print he values we scanned.

AB Amplifier

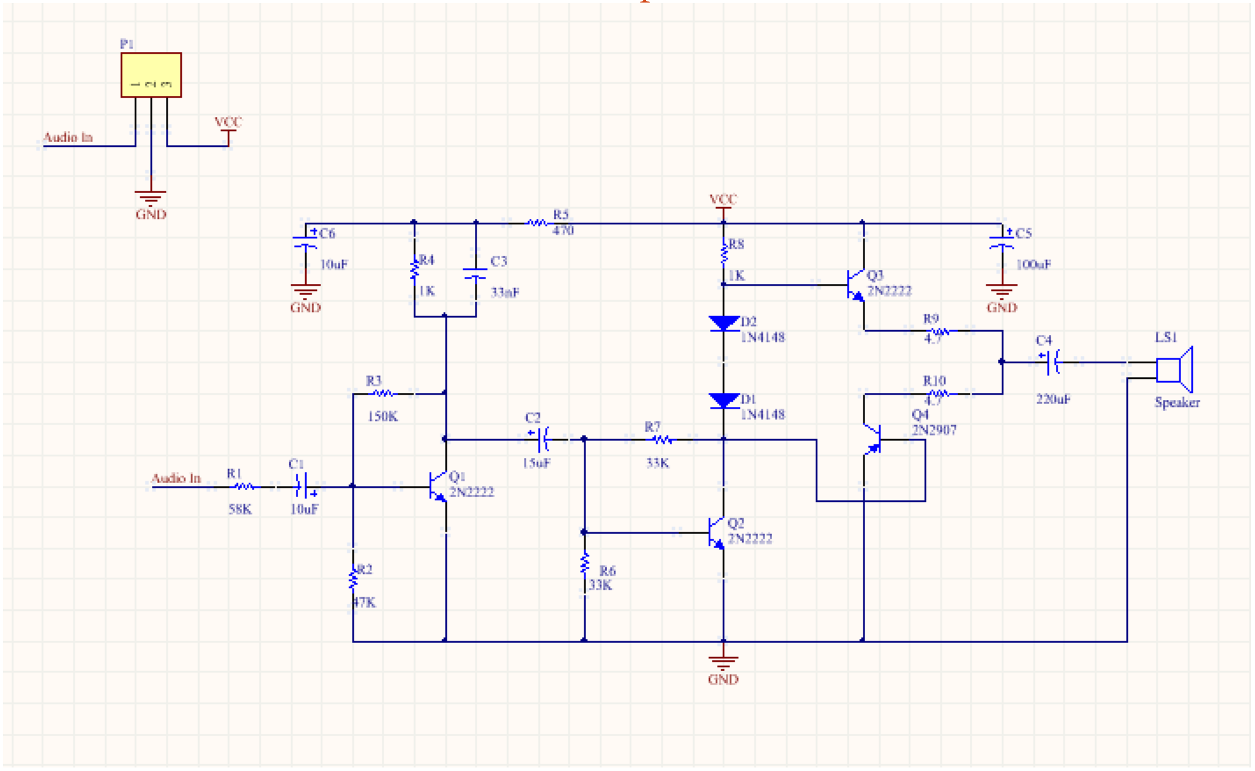


FIGURE 5 AB AMPLIFIER CIRCUIT

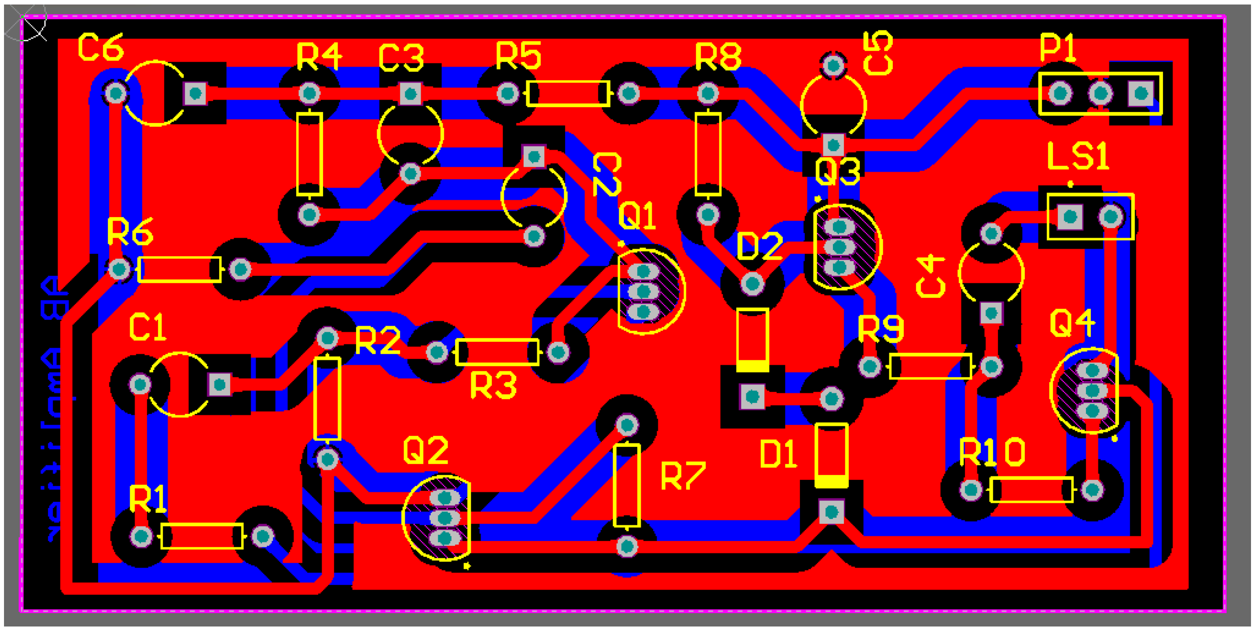
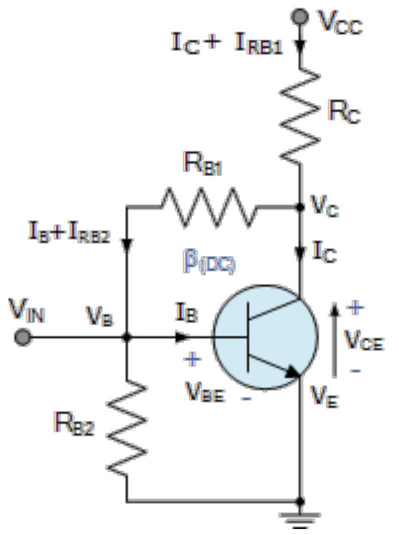


FIGURE 6 AB AMPLIFIER PCB IN ALTIUM DESIGNER

Preamplifier Stage

Transistor Q1 and the surrounding components act as a preamplifier. A preamplifier is a component that provides a small gain to a signal before it is fed to an amplifier with a greater gain. This is done because many audio sources have a small output voltage that would not be enough to drive a speaker. The preamplifier stage also acts as a volume control and noise filter. Transistor Q1 is dual feedback biased using R2, R3, and R4. Biasing sets the transistor DC operating voltage and current to the levels that prevents the AC signal from clipping. It is done because transistors cannot conduct if the voltage/current is negative, which it is in an audio signal. Therefore, the input signal must be shifted into the positive region.
This can be done with collector feedback biasing, also called self-biasing, using R3 and R4. Referring to figure 3, self-biasing works in the following way: If Ic increases, Vc decreases, reducing the base drive and automatically reducing collector current (keeping the Q point fixed). Negative feedback is supplied via Rb1 so that when the load current increases, Vc decreases, causing Ib to decrease, and bringing Ic back to normal. Adding R2 to the circuit makes it dual feedback biased, providing additional stability through automatic biasing and Rf feedback.



$$V_C = V_{CC} - R_C(I_C + I_{B1})$$
$$V_E = 0\text{v}$$
$$V_B = V_{BE}$$
$$I_{RB2} = \frac{V_B}{R_{B2}}$$
$$I_{RB1} = I_B + I_{RB2} = \frac{V_C - V_B}{R_{B1}}$$
$$I_C = \beta_{(DC)} I_B$$
$$I_E = (I_C + I_B) \cong I_C$$

FIGURE 7 DUAL FEEDBACK BIASING

Driver Stage

Q2 acts as the driver stage. The driver stage's purpose is to produce enough current gain to drive the output stage. This in turn causes a gain in power.

AB push pull

Transistors Q3 and Q4 and diodes D1 and D2 form part of the push-pull stage with diode biasing. Temperature can vary the base-emitter voltage. This can be prevented by forward biasing via two diodes. The alternating transistors source or sink the load current, which improves efficiency, increases the power output, cancels even harmonics, and cancels DC current at the output.

Wi-Fi Module

For our Wi-Fi module, we chose the ESP8266, shown below. This module is used most often by hobbyists and beginners due to its short and simple AT commands. It is also one of the cheapest modules on the market, at only \$3.00. Just a few years ago, Wi-Fi modules were priced at over \$50!

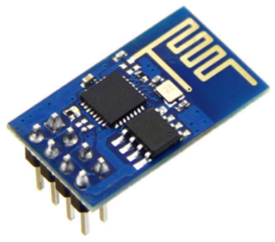
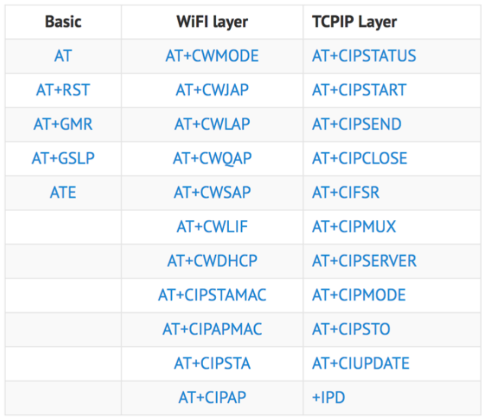


FIGURE 8 THE ESP8266 MODULE



Basic	WiFI layer	TCPIP Layer
AT	AT+CWMODE	AT+CIPSTATUS
AT+RST	AT+CWJAP	AT+CIPSTART
AT+GMR	AT+CWLAP	AT+CIPSEND
AT+GSLP	AT+CWQAP	AT+CIPCLOSE
ATE	AT+CWSAP	AT+CIFSR
	AT+CWLIF	AT+CIPMUX
	AT+CWDHCP	AT+CIPSERVER
	AT+CIPSTAMAC	AT+CIPMODE
	AT+CIPAPMAC	AT+CIPSTO
	AT+CIPSTA	AT+CIUPDATE
	AT+CIPAP	+IPD

FIGURE 1 ALL KNOWN AT COMMANDS

We utilized a mobile hotspot to share access to our phone's wireless network connection with the Wi-Fi module. Wi-Fi utilizes radio waves at a frequency of 2.4 or 5 GHz to establish a connection to a network. Once data is translated into a radio signal, an antenna will transmit to a router. The data is then decoded and sent through the internet via an Ethernet cable. The speed and frequency of transmission are highlighted by the 802.11 network standards a, b, g, and n. Once a connection is established, we send an HTTP request to our server hosted on Amazon Web Services (AWS). HTTP is short for hypertext transfer protocol. When an HTTP request is submitted to a server, the server returns a response message.

A server is a resource provider. It holds HTML files and other content. It can be accessed via a web browser, remote client, or ssh remote login. The server itself is made up of a local client, or a program on the same machine as the server, like MySQL. To pass SQL queries to the server, a programmer must use a scripting language. Initially we considered using JavaScript as our primary scripting language, used solely for web browsers. However, we opted for a simpler general-purpose language called Python.

Connecting to a server also requires an IP address. An Internet Protocol address consists of four numbers ranging from 0 to 255 that designate the destination of the information we send. We also include a port number at which the virtual machine listens in for data.

The Wi-Fi module communicates with the Atmega644PA via UART at 9600 baud. UART0 listens in for the RF ID being sent from the XBEE. The ID is compared to a structure array holding all card values. When there's a match, the microcontroller then decides the status of the dog and prints it on the LCD along with the card ID. Initially, a dog is surrendered and already on the website. Once the card is scanned again, and the dog's status changes, the Wi-FI module sends a GET request—to the server with the card ID and the status in order to override the current status stored in an SQL table. Then python executes the script responsible for changing the dog's picture on the front-end of the website. When a person visits the website, the browser will send a GET request to the server to generate the HTML files.

Final Design Schematics and PCBs

RF Main Board

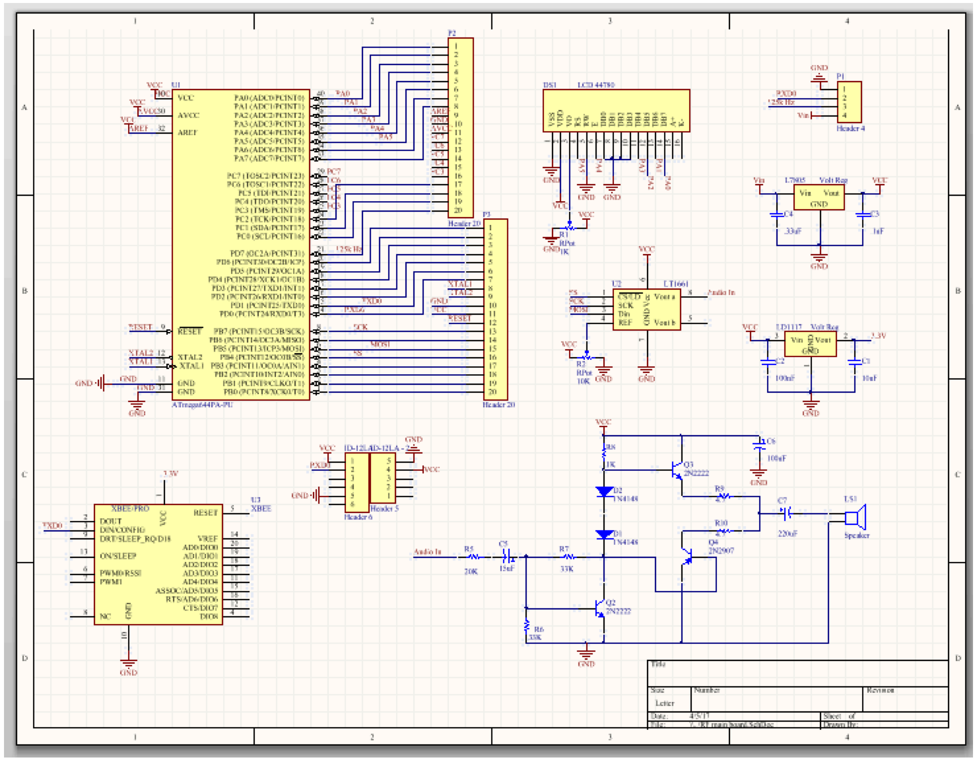


FIGURE 2 RF MAIN BOARD SCHEMATIC

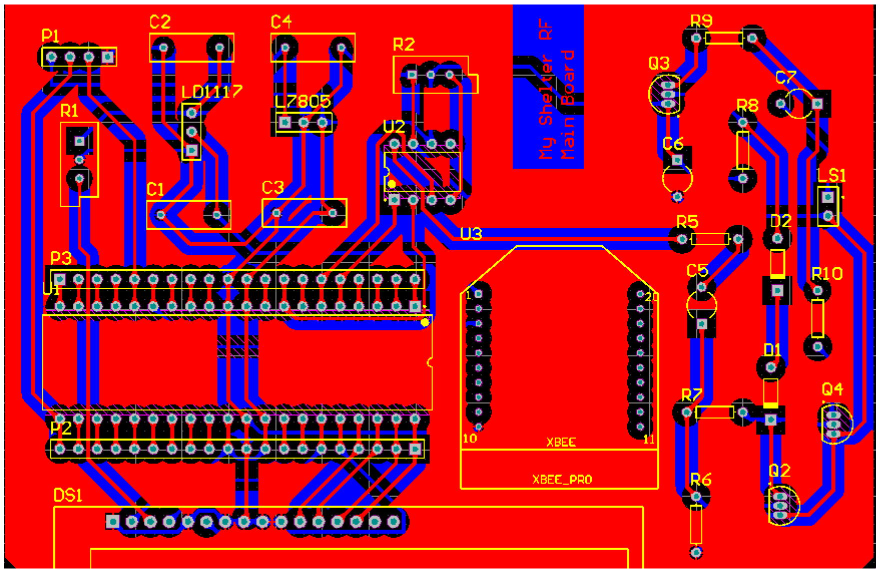


FIGURE 3 RF MAIN BOARD PCB

Wi-FI Main Board

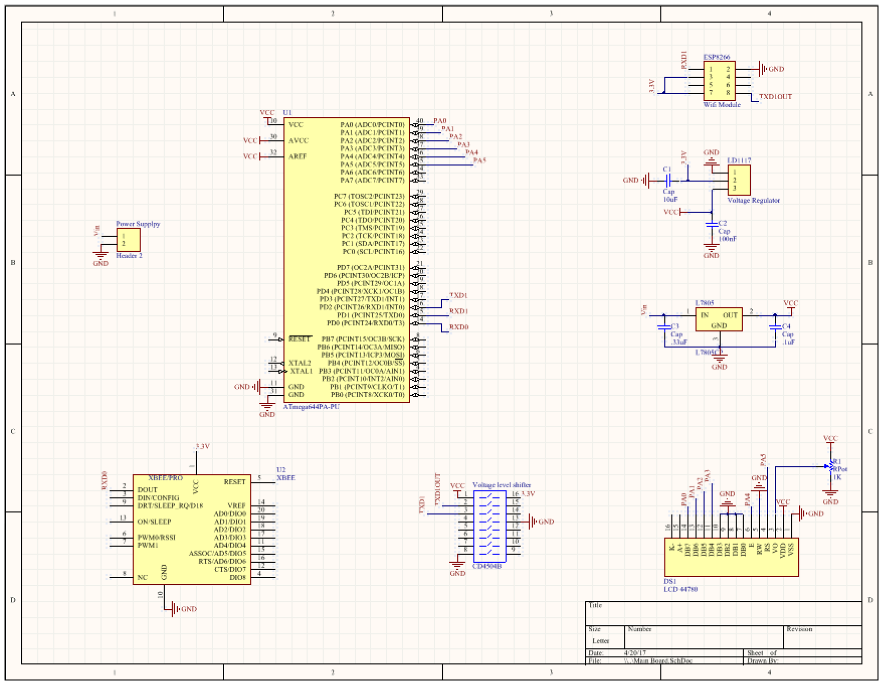


FIGURE 4 WIFI MAIN BOARD SCHEMATIC

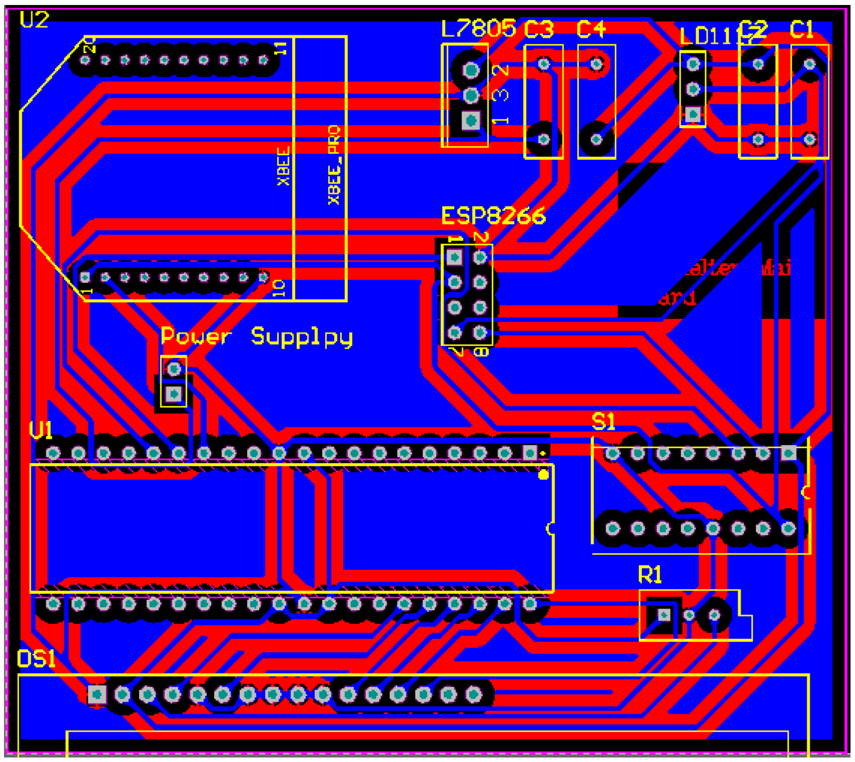


FIGURE 5 WIFI MAIN BOARD PCB

Work Break-Down

Task	**Liz**	**Miguel**
RFID Receiver	0%	100%
Battery Charging Circuit	0%	100%
AB Amplifier	100%	0%
Main Board Programming	100%	0%
Main Board PCB	0%	100%
Database & Website	100%	0%
RF Programming	90%	10%
RF PCB	70%	30%
Case	0%	100%