# Paw Avenue

ECEN 3360 - Digital Design Lab

Clint Olsen

Matt Zarekani

May 11, 2017

# 1 Executive Summary

As we upgrade our homes into smart homes, certain things are left behind. We have upgraded our microwaves, our coffeemakers, and even the locks to our doors. However, it seems that the Internet of Things has left the doggy door behind. Imagine you are a compassionate dog owner with a large backyard. You would like your dog to use the yard even when you're not around to open the door for him. So you install a doggy door. Now you're at work, and it's raining. You dread coming home to see the muddy footprints that are surely all over your beautiful home. Now imagine you have two dogs. One is sick and one is not. The vet has told you not to let the sick dog outside, but you don't want to punish your other dog by locking them both inside. These were the problems which we sought to solve with our smart door. Paw Avenue must be able to do the following things: lock/unlock the dog door based on the dog's distance from the door, track the positions of the dogs, and block/unblock specific dogs based on user input. The user's interaction with our device will be entirely through an easy to use Android application. The dogs will be tracked via Bluetooth Low Energy beacons embedded into their collars. The door will be locked/unlocked via two continuous servos. The door will then communicate all this to the Android app.

# 2 Background

There are a small selection of smart dog doors currently on the market, but all of which use similar components in terms of the way that dogs are detected by the door. The most popular form of dog identification is active and passive RFID collar attachments, used to transmit a unique ID to the door based on the specific dog. In the initial planning of this project, the options of dog detection ranged from the use of passive RFID all the way to pressure sensing mats in front of the doors. For the pressure sensors, there was a clear disadvantage in terms of accuracy and knowing which dog is at the door, so this

idea was ruled out. The next consideration was passive RFID, but after reading up on previous companies' failed attempts to implement this and analyzing the transmission range of these devices in regards to cost effectiveness, they too were ruled out. In order to get the performance needed, i.e a decent range with reasonable cost, Bluetooth Low Energy (BLE) was selected as the transmission mechanism. This differentiates this project from many other implementations because this is first time BLE has been used for a smart dog door implementation. BLE is an emerging form of object detection and is being implemented in many different applications where a "beacon" like situation is required, thus making it feasible and the perfect application for dog distance detection.

The next area that makes this project unique is the implementation of an Android application to control and configure the door itself. Upon research into this area of technology, no other smart dog door technologies seem to implement a mobile app for customization of the behavior of the door and monitoring the connected canines.

These two unique utilizations of BLE and Android applications help create an idea that would be useful and exciting to many individuals. The craze of the Internet of Things and Smart Home technologies is a movement in the world today that is constantly expanding. Bringing peoples' pets into the connected world would just be yet another point of interest for Smart Home fanatics. This door would not just open and close when the dog is near, the full behavior of the door would be customizable using the mobile app. Functionalities would include setting a master lock or unlock for the door, monitoring which dog in inside and outside, and locking out specific dogs if there are multiple being monitored. Take the scenario where one has a new puppy that is not house trained and another, more mature, dog that is house trained. The app would allow you to block the puppy from coming inside based on its unique tag, and allow the other to come and go as it pleases. This is just one of the many scenarios where the integration of Paw Avenue would be an exciting feature to add the tech savvy and dog loving customer.

# 3 Design Methodology

This project consisted of two major technical components: Embedded Systems Design and Android Application design. The specifics of each of these is outlined in the sections below.

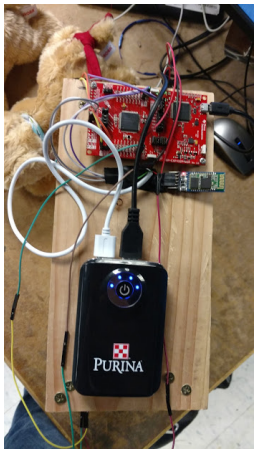
## Embedded Systems Design

### 3.1 The Microcontroller

The basis for the communication between the mobile application, door control mechanisms, and data collection are centered around the implementation of a microcontroller to obtain, process, and respond to data. For this project, the Texas Instruments MSP432p401r was chosen as the center piece of the embedded design. This board was chosen over the LPCXpresso for a variety of reasons. The first reason was the need for 2 UART modules: 1 for communicating with the mobile phone and another for communicating with the Bluetooth Low Energy module, which communicates via UART. The next reason this board was selected was due to the ease of register level programming offered by the structure overlays and naming conventions provided by TI. A final reason this board was chosen was its overall reliability, build quality, and excellent documentation.

### 3.2 Bluetooth Low Energy Receiver Design

In order to receive transmissions from other BLE beacons(more on these later), a central receiver needed to be implemented. For this project, the Sparkfun BLE Mate 2 was selected as the receiver[1]. When using BLE, devices can generally be configured in 2 ways; as a peripheral or as a central device. In order to receive transmissions or connect to other BLE devices, the device must have central mode capabilities. If the device needs to send



out its information to be received by another, the device must be in peripheral mode. The BLE Mate 2 communicates with the microcontroller that it is connected to via UART or Universal Asynchronous Receiver Transmitter. This protocol send data "frames" 1 frame at a time down a physical connection to the other device. A UART device has Rx (receive) and Tx (transmit) terminals that are physically connected to the other device. In the data frame set up of the BLE Mate 2, the frame was very simple, consisting of a start bit, 1 stop bit, and a byte of data bits. So in this case, data is sent to and from the device 1 byte at a time. In addition to this configuration, the UART handling BLE data was also configured to enable receive interrupts, that is when the BLE module receives data from a beacon, and sends it to the microcontroller, the program interrupts and performs special operation. In this case, the device was also configured to send data frames at a baud rate of 9600.

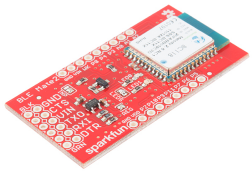


Figure 1: *Sparkfun BLE Mate 2*

In terms of configuring the BLE Mate 2 itself, a built in command set called AT Commands were used to set up the device for the needed functionality. This device came configured with its own set of AT commands that can be seen in the reference section[2]. AT Commands are ASCII encoded commands that control the operation of the device. By sending bytes of ASCII characters to the BLE chip from the microcontroller via UART, the standard configuration needed was configured: central mode on and scan mode on. For this module, as stated before, in order to receive data from others, the device must be configured as a central device. Once in this mode, another mode must be enabled in order to begin scanning for responses from other devices, called scan mode. Therefore, the system overall scans for responses from other devices, sends the received information to the microcontroller via UART, thus interrupting the controllers main program and performs unique functionality.

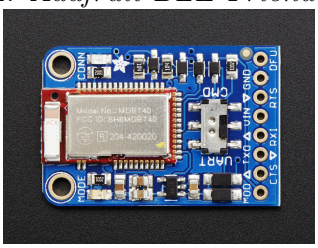
The processing firmware associated with the BLE module parses all of the incoming

beacon transmissions for the data associated with the configured dogs. The microcontroller is constantly receiving and processing data and upon receiving a complete line of information, recognized by the UART receiving a carriage return character, the line is parsed first to see if the line contains the MAC address associated with one of the configured dog collars, and if so locates the transmission power value, or RSSI value. This value is sent as a ASCII representation of a signed 8-bit number, so a conversion is made from ASCII to a int8_t signed value for later use. This value is then checked to see if it falls within the desired transmission power associated with "close to the door" and signals the door to open if the dog is not blocked. Another functionality component is the fact that the previous RSSI values for the other dog is stored so that if that dog is blocked and an unblocked dog comes to the door, it will not open, which keeps unwanted or untrained dogs in or out.

## 3.3 Dog Collar Beacon Devices

At the other end of the transmission spectrum are the BLE beacons that are attached to the collars of each dog to transmit their unique information. For these modules the Adafruit LE UART Friend - Bluetooth Low Energy module was used[3]mainly due to its wide range of voltage operation (3.3V - 15V) which allowed for the use of 2 small CR2032 coin cell batteries to be used for power, adding to the small form factor. These devices came preprogrammed in peripheral mode, so the implementation was much less intensive than with the BLE Mate and out of the box they were ready to be wired and transmit on power up. These devices when powered and in peripheral mode send packets of information regarding itself, such as its unique MAC address and RSSI value. Both of these key pieces of information will be utilized to configure the dogs uniquely and tell how close the dog is to the receiver on the door.

Figure 2: *Adafruit BLE Friend Module*




## 3.4 Locking Mechanism

The next portion of the design includes the mechanism that is used to lock the door itself. The chosen prototype design consisted of using two servo motors, each with a wooden plank attached, located on both sides of the door. Upon receiving an appropriate command, the planks would lower down to straddle the door on each side, preventing it from moving. The servos themselves were the SM-S4303R high torque continuous rotation servo, also from Sparkfun[4]. To power the servos and the board itself, a rechargeable USB power bank with two 5V outputs was used due to the fact the servos and MSP can all operate at 5V. The rotation of the servos functions based off of a pulse width modulation signal at a specific pulse width. For this servo, a pulse width greater than 1.5 ms correlated to one direction of rotation and less than 1.5 ms correlated to the other direction. An easy way to achieve a change in rotation is to change the duty cycle of a particular frequency in order to quickly change the pulse width. To provide this PWM signal to the servos, the timer and GPIO pins on the MSP were utilized. The 32 bit timer of the MSP was configured to interrupt every time it counted to a particular value (capture compare interrupts), which in this case correlated to interrupts every millisecond. Utilizing this convenient interrupt time, the interrupt service routine counts the number of milliseconds that had passed in order to toggle an output pin at the correct frequency and duty cycle using modular arithmetic (see main.c for timer ISR). This method was duplicated onto another pin such that both servos could receive this signal.

### 3.5 Bluetooth Classic for Communication with Smartphone

Having multiple UART capability allowed this project to become more expansive and functional by allowing the integration of the BLE input and also input from a smartphone. The second UART in this case was connected to the HC-05 Bluetooth module, which is a simple UART Bluetooth receiver transmitter[5]. As with the BLE module, the UART that handles this device was also set up with same data frame parameters, baud rate, and receive interrupts for immediate processing of incoming data. The microcontroller was configured to respond to incoming data based on a simple command set from the mobile application. For example, when the device was commanded to lock or unlock, the module would receive a 'L' or 'U' 8-bit character which would be recognized by the microcontroller, responding by activating the servos. There are also built in safeguards that prevent the device from performing multiple operations in a row, such as unlocks or locks, if not desired. Another feature that is integrated is the continual updating of status between the two devices if disconnected. Upon connection to the Bluetooth module, the phone will send a special command, indicating that it needs an up to date status of the door so the two components remain synchronized. This status update includes parameters such as which dogs are inside or outside and if the door is locked or unlocked.

## Android Application Design

The goal of the app was to be the one stop destination for interacting with Paw Avenue. It was therefore essential for the app to be clean and user friendly, but at the same time comprehensive in its functionality. In order to communicate with the the Bluetooth module we are using the Bluetooth code provided in class as a foundation. Figure 3 shows the application's home page.

## 3.6 Lock/Unlock

The Lock and Unlock buttons do just that. The door image in the bottom half of Figure 3 toggles between open and closed based on the lock/unlock status. In order to avoid a mismatch between the door and the app, the app only switches its position upon receiving a confirmation command from the Bluetooth module. Much of the app works in the same way: variables which hold a positional value do not change until a confirmation command is sent back from the door.

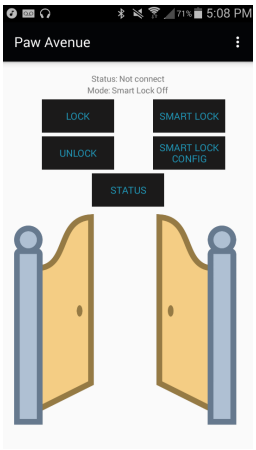



Figure 3: Home Page

## 3.7 Smart Lock

The "Smart Lock" button toggles the smart lock functionality which is based around distance detection of the dogs and their blocked status. What happens here is based on the settings in "Smart Lock Config" shown in Figure 4. Here we our two dogs: Flash and Sparky. Under their names they each have a drop down menu (called spinners in Android Studio). In the drop down menu the user can decide whether to allow that dog to access the doggy door or not. Of course this is only relevant when the "Smart Lock" feature is turned on. When the user is satisfied with their selection they can click save and will be returned to the home page. On clicking save the app will check for an active connection with the Bluetooth module and if one is found it will save the user's settings to a file and inform the module of the change in settings. The saved file will be loaded each time the activity is created or resumed.

## 3.8 Status

A responsible pet owner will want to know where her dogs are at all times. Our "Status" page(as seen in Figure 5) will let her know just that. As a canine exits or enters

through our Smart Door the Bluetooth module will inform the app in both Smart Lock mode and normal mode. Should the app not be connected at the time then the module will simply update the app the next time it connects. Unfortunately, our smart door is not perfect, and at times it will think a dog is outside when it is really inside, and vice versa. To remedy this, we have allowed the user to overwrite the current location of the canines. Like the "Smart Lock Config" page the "Status" page has a drop down menu for overwriting the dogs' positions. The "Refresh" button will update the dogs' positions should there be anything to update.

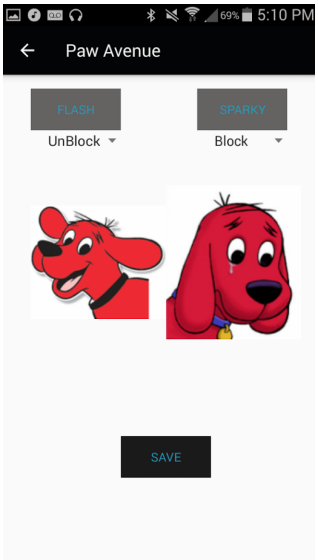



Figure 4: Smart Lock Config

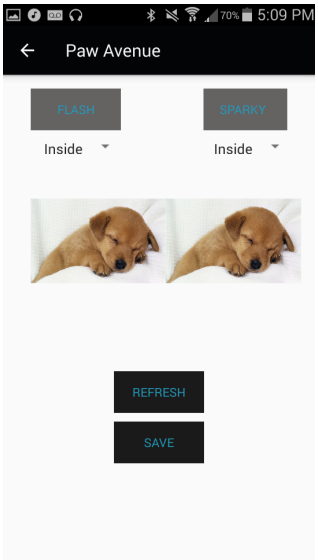



Figure 5: Dog Status

# 4 Results and Discussions

The overall testing of this project came in several different parts with several complications. The largest testing aspect consisted of getting the BLE to function properly. Initially, only the Adafruit BLE modules were purchased with the thought that they could be configured in both peripheral and central mode easily. After receiving the devices and reading through the datasheets more throughly, the devices were clearly not permitted to

change modes in hopes of providing simple integration with Arduino microcontrollers. This led to thoughts of changing direction entirely and switching to RFID instead because of difficulties finding a module that could be configured in central mode. However, after more research into BLE, the BLE Mate was discovered as a reasonably priced solution that could be configured. Upon receiving the board, testing began as a simple program to send and receive responses to the board via UART. The board was configured to send 'OK' or 'ERR' based on the success of the AT commands, so after toying with the syntax of the commands, a small command library was built to easily send commands (as seen in the uart.c source file below). From here, the next challenge was receiving data from the beacons and printing it out to the screen to see. At this point, it had been determined that the Adafruit boards came configured to beacon by default, but it was unknown if the BLE Mate was compatible. After configuring the BLE Mate to central and scan mode, no data was being received, which was concerning. However, after more research it was determined that the scanning parameters were likely the issue. There is a setting that allows for configuration of the scan window and interval, as in how long it scans and how frequently that scan occurs. To test to see if this was the issue, the window was set to scan for several seconds at a similarly long interval. Sure enough, this started producing results. The interval was likely too short by default, causing receive abnormalities in the UART module. With device responses now being received, a Android app (nRF Connect) was used to determine the MAC address of our beacon, and from here the BLE Mate was showing valid responses from the Adafruit boards.

The next major portion of testing included placement of the BLE Mate on the door itself for maximizing transmission consistency with the beacons. The initial placement of the board was leading to large inconsistencies in received transmission power, which caused the door to open even when the dogs were far away from the receiver. After tying many clever software schemes to rectify the issue, it was clear that there was a more inherent issue, which just turned out to be the placement of the device. Upon moving the module at a more even

level with where the dogs would be, the device was very consistent and accurate to detect the locations of the dogs. From here, the transmission power levels were logged numerous times to determine the best threshold for when the door should open and strangely, for both modules, the sweet spot was values greater than -50 dB, which continually showed up when the dog was about a foot from the door, so this value was used with great success.

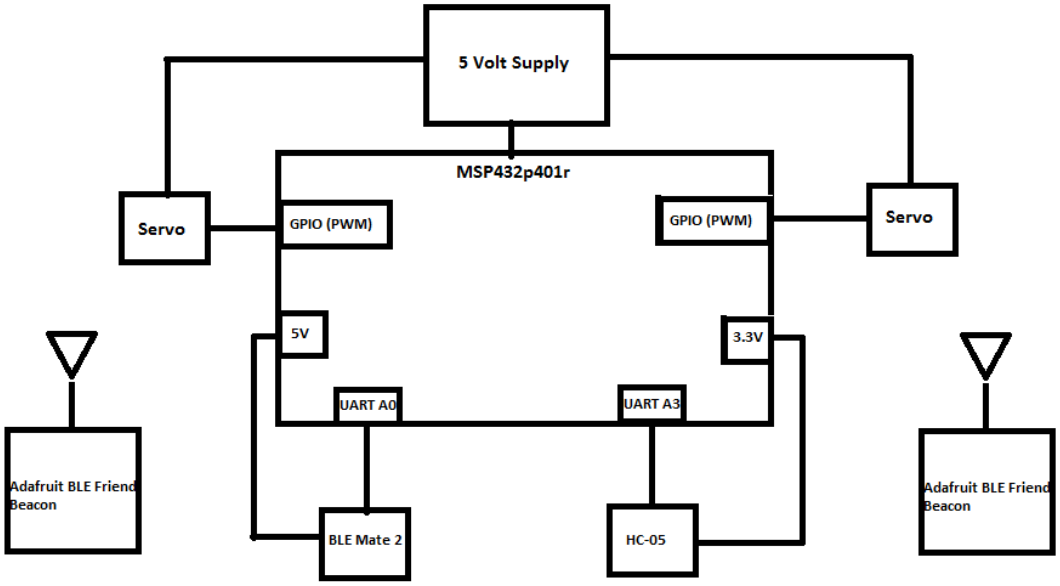
Overall, the testing for the UARTS, PWM, and smartphone communication were all relatively straightforward due to their previous extensive use.

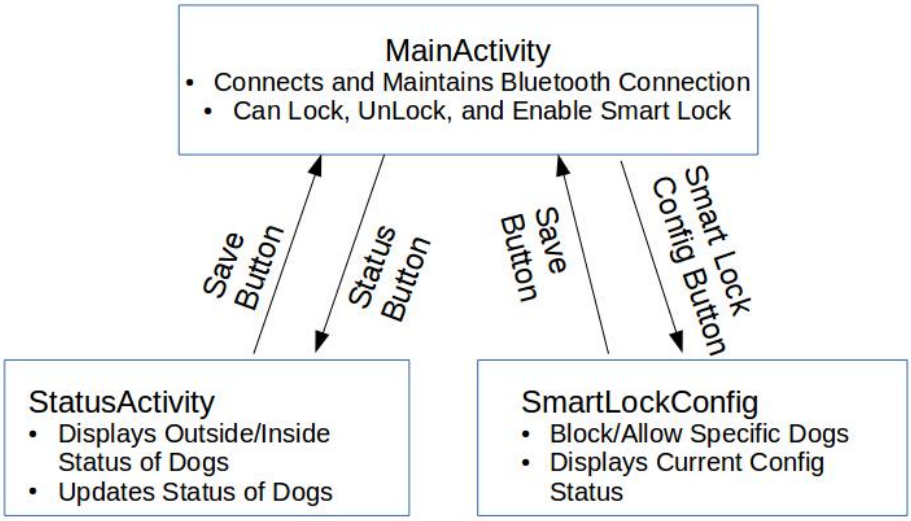
The project in its entirety had a cost of about $100 including the BLE sensors, the MSP, servos, wood materials, and added aesthetic components. In terms of overall workload, this project took between 40 and 50 hours of testing, programming, and physical construction to complete. With this amount of work, the project turned out to be a pretty big success. After the changing of placement of the BLE central module, there was very fast response to the location of the dog, which made it more practical in application. Another aspect of success was the UI created for the app, which had fast and reliable communication with the door itself and a nice aesthetic component for easily configuring the door. If given more time, however, there are still many improvements to be made. One of the biggest improvements would be the handling of very crowded BLE airspace. It was noticed that in the presence of lots of BLE devices (like many phones) the module was much less responsive. This is likely due to the module being flooded with other device beacons, reducing the chance of recognizing the dog collars. This would be a necessary area of improvement, especially if the door is to be integrated into a Smart Home with lots of potential interference. Another area would be the structural design overall. The locking mechanism in particular was quite insecure and crude, so in a non-prototype, this would need to be improved upon to withstand dogs pushing on it, as well as wind and other factors. As a next pass for the door itself, it would be nice to 3D print the entire structure to enclose all of the components and provide a more appealing look in general with a form factor that would be feasible for easy installation in homes.

# 5 Conclusions

For our final project we built an automatic dog door. We used BLE to detect the location of the dogs, and we built an Android application for the user interface. The finished state of our project preformed to our expectations when we tested it. That is to say, it detected the location of the dogs with reasonable accuracy and speed. However, during our demo we experienced a slow detection rate. We believe this was due to the large number of Bluetooth devices that were present at this time. This makes sense when we consider how we were reading scanned BLE devices,i.e. we read them as we got them. We throughly enjoyed working on this project. It was the iterative conclusion of everything that we had learned in this course with some additional things like BLE. We also saw a lot of great ideas from our peers during demo day. We were really impressed by the motion detector video game, the frequency dependent LED project, and the heart rate monitor. Unfortunately, it was hard to hear and see some of the presentations which was an issue related to our main concern in the class: the lab space is too crowded. However, the class was an otherwise great learning experience.

# Flow and block-level diagrams



# References

[1] https://www.sparkfun.com/products/13019

[2] https://cdn.sparkfun.com/datasheets/Wireless/Bluetooth/Melody-Smart-V2_6_0-UserManual-Rev_E.pdf

[3] https://www.adafruit.com/product/2479

[4] https://www.sparkfun.com/products/9347

[5] https://arduino-info.wikispaces.com/BlueTooth-HC05-HC06-Modules-How-To

[6] https://github.com/akexorcist/Android-BluetoothSPPLibrary

# Code

## MSP Code: main.c

```
/*
 * Clint Olsen and Matt Zarekani
 * ECEN 3360
 * Final Project: Automatic Dog Door
 *
 */

#include "msp.h"
#include "uart.h"
#include "timer.h"
#include "gpio.h"

#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>

//Interrupt handlers
void EUSCIA0_IRQHandler(void);
void EUSCIA3_IRQHandler(void);
void TA0_0_IRQHandler(void);

volatile uint32_t period = 10; //time in milliseconds
volatile uint32_t duty_cycle = 10; //i.e. 50 indicates 50% duty cycle
volatile uint32_t timerA0_counter = 0;
volatile uint32_t temp = 0; //for timer


//door variables
uint8_t smartLockOn = 0;
uint8_t locked = 0;
uint8_t unlocked = 1;
uint8_t servoOn = 0;

//dog variables
uint8_t dog1Blocked = 0;
uint8_t dog2Blocked = 0;
uint8_t dog1Inside = 1;
uint8_t dog2Inside = 1;
uint8_t changeDog1 = 0;
uint8_t changeDog2 = 0;
volatile int8_t rssiDog1Prev = -80;
volatile int8_t rssiDog2Prev = -80;
extern volatile uint8_t rssiPassDog1;
extern volatile uint8_t rssiPassDog2;


int counter = 0;
const int size = 400;

int startCollection = 0;
```

```
uint8_t response[size];

int done = 0;


void main(void)
{
    WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer

    //Peripheral configuration
    configure_serial_portA0();
    configure_serial_portA3();
    configure_pins();
    configure_timer();

    centralOn();
    scanOn();

    int i;
    while(1){
        //if a full string was recieved, check for the correct addresses
        if(done == 1){
                checkBLEResponse(response, counter);
                //In this case we want BLE to open the dogs only for specific dogs
                if(rssiPassDog1 == 1 && dog1Blocked == 0 && smartLockOn == 1){
                        //scanOff();
                        if(dog1Inside == 1){
                                //dog 1 went outside
                                dog1Inside = 0;
                                uart_putcharA3('1');
                                uart_putcharA3('O');
                                uart_putcharA3('\r');
                        }
                        else{
                                //dog 1 went inside
                                dog1Inside = 1;
                                uart_putcharA3('1');
                                uart_putcharA3('I');
                                uart_putcharA3('\r');
                        }
                        rssiPassDog1 = 0;
                        duty_cycle = 10; //set unlock duty cycle
                        unlocked = 1;
                        locked = 0;
                        servoOn = 1;
                        TA0CCTL0 |= TIMER_A_CCTLN_CCIE; // TACCR0 interrupt enabled
                        for(i = 0; i < 1000000; i++); //delay for dog to walk through
                        servoOn = 1; //indicate servo turn to timer
                        duty_cycle = 20; //set lock duty cycle
                        unlocked = 0;
                        locked = 1;
                        TA0CCTL0 |= TIMER_A_CCTLN_CCIE; // TACCR0 interrupt enabled
                }
                else if(rssiPassDog2 == 1 && dog2Blocked == 0 && smartLockOn == 1){
                                //scanOff();
```

```
                        if(dog2Inside == 1){
                                //dog 2 went outside
                                dog2Inside = 0;
                                uart_putcharA3('2');
                                uart_putcharA3('O');
                                uart_putcharA3('\r');
                        }
                        else{
                                //dog 2 went inside
                                dog2Inside = 1;
                                uart_putcharA3('2');
                                uart_putcharA3('I');
                                uart_putcharA3('\r');
                        }
                rssiPassDog2 = 0;
                duty_cycle = 10; //set unlock duty cycle
                unlocked = 1;
                locked = 0;
                servoOn = 1;
                TA0CCTL0 |= TIMER_A_CCTLN_CCIE; // TACCR0 interrupt enabled
                for(i = 0; i < 1000000; i++); //delay for dog to walk through
                servoOn = 1; //indicate servo turn to timer
                duty_cycle = 20; //set lock duty cycle
                unlocked = 0;
                locked = 1;
                TA0CCTL0 |= TIMER_A_CCTLN_CCIE; // TACCR0 interrupt enabled
        }
else if(rssiPassDog1 == 1 && smartLockOn == 0){
        //send command to say dog went in or out
        rssiPassDog1 = 0;
        if(dog1Inside == 1){
                //dog 1 went outside
                dog1Inside = 0;
                uart_putcharA3('1');
                uart_putcharA3('O');
                uart_putcharA3('\r');
        }
        else{
                //dog 1 went inside
                dog1Inside = 1;
                uart_putcharA3('1');
                uart_putcharA3('I');
                uart_putcharA3('\r');
        }
        for(i = 0; i < 500000; i++); //delay for dog to walk through

}
else if(rssiPassDog2 == 1 && smartLockOn == 0){
        //send command to say dog went in or out
        rssiPassDog2 = 0;
        if(dog2Inside == 1){
                //dog 2 went outside
                dog2Inside = 0;
                uart_putcharA3('2');
                uart_putcharA3('O');
                uart_putcharA3('\r');
```

```
                        }
                        else{
                                //dog 2 went inside
                                dog2Inside = 1;
                                uart_putcharA3('2');
                                uart_putcharA3('I');
                                uart_putcharA3('\r');
                        }
                        for(i = 0; i < 500000; i++); //delay for dog to walk through
                }
                counter = 0;
                done = 0;
                //centralOn();
                //scanOn();
                UCA0IE |= EUSCI_A__RXIE; // Enable USCI_A0 RX interrupts
        }
    }
}
// BLE communication
void EUSCIA0_IRQHandler(void){
        uint8_t data;

        //recieve interrupts
        if (UCA0IFG & UCRXIFG){
                //code to handle RX interrupts
                //prints recived data back out to terminal
                data = UCA0RXBUF;
                if(data == '\r'){
                        //printf("r");
                        response[counter] = data;
                        done = 1;
                        UCA0IE &= ~(EUSCI_A__RXIE); // Disable USCI_A0 RX interrupts

                }
                else{
                        response[counter] = data;
                        counter++;
                        }
        }
        //transmit interrupts
        if (UCA0IFG & UCTXIFG){

        }
}

//For Bluetooth module to talk to Andorid App
void EUSCIA3_IRQHandler(void){
        uint8_t data;

        //recieve interrupts
        if (UCA3IFG & UCRXIFG){
                //code to handle RX interrupts
                //prints recived data back out to terminal
                data = UCA3RXBUF;
                //lock on command from smart phone
                if(smartLockOn == 0 && data == 'L'){
```

```
        if(locked == 0){
                uart_putcharA3('L');
                uart_putcharA3('\r');
                locked = 1;
                servoOn = 1;
                unlocked = 0;
                duty_cycle = 20;
                TA0CCTL0 |= TIMER_A_CCTLN_CCIE; // TACCR0 interrupt enabled
        }
}
//lock on command from smart phone
else if(smartLockOn == 0 && data == 'U'){
        if(unlocked == 0){
                uart_putcharA3('U');
                uart_putcharA3('\r');
                unlocked = 1;
                servoOn = 1;
                locked = 0;
                duty_cycle = 10;
                TA0CCTL0 |= TIMER_A_CCTLN_CCIE; // TACCR0 interrupt enabled
        }

}
//lock on command from smart phone
else if(data == 'S'){
        if(locked == 0 && smartLockOn == 0){
                smartLockOn = 1;
                servoOn = 1;
                locked = 1;
                unlocked = 0;
                duty_cycle = 20;
                TA0CCTL0 |= TIMER_A_CCTLN_CCIE; // TACCR0 interrupt enabled
            centralOn();
            scanOn();
        }
        else if(locked == 1 && smartLockOn == 0){
                smartLockOn = 1;
        }
        else if(smartLockOn == 1){
                smartLockOn = 0;
        }

}
else if(data == 'C'){
        if(dog1Inside == 1){
                uart_putcharA3('1');
                uart_putcharA3('I');
                uart_putcharA3('\r');

        }
        else{
                uart_putcharA3('1');
                uart_putcharA3('O');
                uart_putcharA3('\r');

        }
```

```
                if(dog2Inside == 1){
                        uart_putcharA3('2');
                        uart_putcharA3('I');
                        uart_putcharA3('\r');

                }
                else{
                        uart_putcharA3('2');
                        uart_putcharA3('O');
                        uart_putcharA3('\r');

                }
        }
        else if(changeDog1 == 1 && data == 'A'){
                dog1Blocked = 0;
                changeDog1 = 0;
        }
        else if(changeDog1 == 1 && data == 'B'){
                dog1Blocked = 1;
                changeDog1 = 0;
        }
        else if(changeDog1 == 1 && data == 'I'){
                dog1Inside = 1;
                changeDog1 = 0;
        }
        else if(changeDog1 == 1 && data == 'O'){
                dog1Inside = 0;
                changeDog1 = 0;
        }
        else if(changeDog2 == 1 && data == 'A'){
                dog2Blocked = 0;
                changeDog2 = 0;
        }
        else if(changeDog2 == 1 && data == 'B'){
                dog2Blocked = 1;
                changeDog2 = 0;
        }
        else if(changeDog2 == 1 && data == 'I'){
                dog2Inside = 1;
                changeDog2 = 0;
        }
        else if(changeDog2 == 1 && data == 'O'){
                dog2Inside = 0;
                changeDog2 = 0;
        }
        else if(data == '1'){
                changeDog1 = 1;
        }
        //change dog 2 settings
        else if(data == '2'){
                changeDog2 = 1;
        }
        printf("%c", data);

}
//transmit interrupts
```

```
        if (UCA3IFG & UCTXIFG){

        }
}

//PWM output for servos
void TA0_0_IRQHandler(void) {
        //turn off timer interrupt
        TA0CCTL0 &= ~CCIFG;

        timerA0_counter++;

        //PWM production from Port 2 Pin 5 and Port 3 Pin 0
        if (servoOn == 1 && timerA0_counter % period == 0) {
                temp = timerA0_counter; //store the time a new period starts
                P2OUT |= BIT5;
                P3OUT |= BIT0;
        } else if (servoOn == 1 && timerA0_counter == (temp + (uint32_t) ((double) period * (double)
             duty_cycle/ (double) 100))) {
                P2OUT &= ~BIT5;
                P3OUT &= ~BIT0;

        }
        //wait for lock duration
        if(servoOn == 1 && timerA0_counter >= 200 && locked == 1){
                TA0CCTL0 &= ~TIMER_A_CCTLN_CCIE; // TACCR0 interrupt enabled
                timerA0_counter = 0;
                servoOn = 0;
        }
        //wait for unlock duration
        if(servoOn == 1 && timerA0_counter >= 250 && unlocked == 1){
                TA0CCTL0 &= ~TIMER_A_CCTLN_CCIE; // TACCR0 interrupt enabled
                timerA0_counter = 0;
                servoOn = 0;
        }

}
```

# MSP Code: uart.h

```
/*
 * Clint Olsen and Matt Zarekani
 * ECEN 3360
 * Final Project: Automatic Dog Door
 *
 */

#ifndef UART_H_
#define UART_H_

#include <stdint.h>


void configure_serial_portA0(void);
void configure_serial_portA3(void);
```

```
void uart_putchar_nA0(uint8_t * data, uint32_t length);
void uart_putcharA0(uint8_t tx_data);
void checkBLEResponse(uint8_t *response, int counter);
void uart_putcharA3(uint8_t tx_data);


//BLE AT Commands
void restore();
void storeChanges();
void scanDog1();
void setScanTimeout();
void setScanInterval();
void cconOff(void);
void connect();
void reset();
void centralOff();
void centralOn();
void scanOff();
void scanOn();
void config(void);
void status(void);
void advOff(void);
void advOn(void);


#endif /* UART_H_ */
```

## 5.1 MSP Code: uart.c

```
/*
 * Clint Olsen and Matt Zarekani
 * ECEN 3360
 * Final Project: Automatic Dog Door
 *
 */
#include "uart.h"
#include "msp.h"
#include <stdint.h>

//UART preprocessor directives
//Baud Rate
#define br0 0x13 //9600 Baud
#define br1 0x00


#define CR 0x0D //carriage return

volatile uint8_t rssiPassDog1 = 0;
extern volatile int8_t rssiDog1Prev;
extern volatile int8_t rssiDog2Prev;
volatile uint8_t rssiPassDog2 = 0;


extern uint8_t dog1Blocked;
extern uint8_t dog2Blocked;
```

```
//Configure the serial port for the reciever specifications (BLE Module)
void configure_serial_portA0(){
        UCA0CTLW0 |= UCSWRST;       // Put eUSCI in reset
        P1SEL0 |= BIT2;                         //Rx Primary mode
        P1SEL1 &= ~BIT2;

        P1SEL0 |= BIT3;                         //Tx Primary mode
        P1SEL1 &= ~BIT3;
        // Select Frame parameters and clock source
        UCA0CTLW0 |= EUSCI_A_CTLW0_SSEL__SMCLK;
        UCA0CTLW0 &= ~EUSCI_A_CTLW0_PEN; //parity disabled
        UCA0CTLW0 &= ~EUSCI_A_CTLW0_MODE0; // set to uart mode
        UCA0CTLW0 &= ~EUSCI_A_CTLW0_MODE1;
        UCA0CTLW0 &= ~EUSCI_A_CTLW0_MSB;//LSB first
        UCA0CTLW0 &= ~EUSCI_A_CTLW0_SEVENBIT;//8 bit character length
        UCA0CTLW0 &= ~EUSCI_A_CTLW0_SPB; //one stop bit one start bit is default
        UCA0MCTLW |= EUSCI_A_MCTLW_OS16;
        UCA0MCTLW |= 0x0080;
        UCA0MCTLW &= 0x00FF;
        UCA0BR0 = br0;     // Set Baud Rate
        UCA0BR1 = br1;
        UCA0CTLW0 &= ~UCSWRST;       // Initialize eUSCI
        UCA0IE |= EUSCI_A__RXIE; // Enable USCI_A0 RX interrupts


        NVIC->ISER[0] = 1 << ((EUSCIA0_IRQn) & 31);
}

//Configure the serial port for the reciever specifications (HC-05 for User input from app)
void configure_serial_portA3(void){
        UCA3CTLW0 |= UCSWRST;       // Put eUSCI in reset
        P9SEL0 |= BIT6;                         //Rx Primary mode
        P9SEL1 &= ~BIT6;

        P9SEL0 |= BIT7;                         //Tx Primary mode
        P9SEL1 &= ~BIT7;
        // Select Frame parameters and clock source
        UCA3CTLW0 |= EUSCI_A_CTLW0_SSEL__SMCLK;
        UCA3CTLW0 &= ~EUSCI_A_CTLW0_PEN; //parity disabled
        UCA3CTLW0 &= ~EUSCI_A_CTLW0_MODE0; // set to uart mode
        UCA3CTLW0 &= ~EUSCI_A_CTLW0_MODE1;
        UCA3CTLW0 &= ~EUSCI_A_CTLW0_MSB;//LSB first
        UCA3CTLW0 &= ~EUSCI_A_CTLW0_SEVENBIT;//8 bit character length
        UCA3CTLW0 &= ~EUSCI_A_CTLW0_SPB; //one stop bit one start bit is default
        UCA3MCTLW |= EUSCI_A_MCTLW_OS16;
        UCA3MCTLW |= 0x0080;
        UCA3MCTLW &= 0x00FF;
        UCA3BR0 = br0;     // Set Baud Rate
        UCA3BR1 = br1;
        UCA3CTLW0 &= ~UCSWRST;       // Initialize eUSCI
        UCA3IE |= EUSCI_A__RXIE; // Enable USCI_A0 RX interrupts


        //NVIC->ISER[0] = 1 << ((EUSCIA0_IRQn) & 31);
```

```
        NVIC_EnableIRQ(EUSCIA3_IRQn);

}


//sends a single 8 bit value to the tx buffer
void uart_putcharA0(uint8_t tx_data){
        while(!(UCA0IFG & UCTXIFG));// Blockuntil transmitter is ready
        UCA0TXBUF = tx_data;     // Load data onto buffer
}

//sends a single 8 bit value to the tx buffer
void uart_putcharA3(uint8_t tx_data){
        while(!(UCA3IFG & UCTXIFG));// Blockuntil transmitter is ready
        UCA3TXBUF = tx_data;     // Load data onto buffer
}

//sends a array of 8 bit values to tx buffer
void uart_putchar_nA0(uint8_t * data, uint32_t length){
        // Code to iterate through transmit data
        uint32_t i;
        for(i = 0; i < length; i++){
                uart_putcharA0(data[i]);
        }
}

//checks each line of scanned input to find the addresses of the dog modules
void checkBLEResponse(uint8_t *response, int counter){
        int8_t rssi;
        int i;
        if(response[0] == 'S'){
                //check dog 1 address
                if(response[6] == 'C' && response[10] == '9' && response[17] == '6'){
                        if(response[26] >= (uint8_t)65 && response[26] <= (uint8_t)70){
                                        rssi = response[26] - (uint8_t)55;
                        }
                    else if(response[26] >= (uint8_t)48 && response[26] <= (uint8_t)57){
                        rssi = response[26] - (uint8_t)48;
                    }
                    if(response[25] >= (uint8_t)65 && response[25] <= (uint8_t)70){
                        rssi |= ((response[25] - (uint8_t)55) << 0x04);
                    }
                    else if(response[25] >= (uint8_t)48 && response[25] <= (uint8_t)57){
                                rssi |= ((response[25] - (uint8_t)48) << 0x04);

                        }
                    //prints for testing
                                /*for(i = 0; i < counter; i++){
                        printf("%c", response[i]);
                     }

                   printf("RSSI: %d", rssi);
                   printf("\n");*/
                   rssiDog1Prev = rssi;
                   //allow through if the rssi is valid and other dog is far away
```

```
                        if(rssi > -50 && rssiDog2Prev < -60){
                                rssiPassDog1 = 1;
                        }
                        //allow through if the rssi is valid and other dog is allowed
                        else if(rssi > -50 && dog2Blocked == 0){
                                rssiPassDog1 = 1;
                        }
                }
                        //check dog 2 address
                if(response[6] == 'D' && response[10] == '8' && response[17] == '6'){
                        if(response[26] >= (uint8_t)65 && response[26] <= (uint8_t)70){
                                        rssi = response[26] - (uint8_t)55;
                        }
                else if(response[26] >= (uint8_t)48 && response[26] <= (uint8_t)57){
                        rssi = response[26] - (uint8_t)48;
                }
                if(response[25] >= (uint8_t)65 && response[25] <= (uint8_t)70){
                        rssi |= ((response[25] - (uint8_t)55) << 0x04);
                }
                else if(response[25] >= (uint8_t)48 && response[25] <= (uint8_t)57){
                                rssi |= ((response[25] - (uint8_t)48) << 0x04);

                        }
                //prints for testing
                                /*for(i = 0; i < counter; i++){
                        printf("%c", response[i]);
                 }
                        printf("RSSI: %d", rssi);
                        printf("\n");*/
                        rssiDog2Prev = rssi;
                        //allow through if the rssi is valid and other dog is far away
                        if(rssi > -50 && rssiDog1Prev < -60){
                                rssiPassDog2 = 1;
                        }
                        //allow through if the rssi is valid and other dog is allowed
                        else if(rssi > -50 && dog1Blocked == 0){
                                rssiPassDog2 = 1;
                        }
                }
            }
}

//BLE AT Command Set
void advOn(void){
        uart_putcharA0('A');
        uart_putcharA0('D');
        uart_putcharA0('V');
        uart_putcharA0('_');
        uart_putcharA0('O');
        uart_putcharA0('N');
        uart_putcharA0('\r');
}

void advOff(void){
        uart_putcharA0('A');
        uart_putcharA0('D');
```

```
        uart_putcharA0('V');
        uart_putcharA0('_');
        uart_putcharA0('O');
        uart_putcharA0('F');
        uart_putcharA0('F');
        uart_putcharA0('\r');
}

void status(void){
    uart_putcharA0('S');
    uart_putcharA0('T');
    uart_putcharA0('S');
    uart_putcharA0('\r');
}

void config(void){
    uart_putcharA0('C');
    uart_putcharA0('F');
    uart_putcharA0('G');
    uart_putcharA0('\r');
}

void scanOn(){
        uart_putcharA0('S');
        uart_putcharA0('C');
        uart_putcharA0('N');
        uart_putcharA0('_');
        uart_putcharA0('O');
        uart_putcharA0('N');
        uart_putcharA0('\r');
}

void scanOff(){
        uart_putcharA0('S');
        uart_putcharA0('C');
        uart_putcharA0('N');
        uart_putcharA0('_');
        uart_putcharA0('O');
        uart_putcharA0('F');
        uart_putcharA0('F');
        uart_putcharA0('\r');
}

void centralOn(){
        uart_putcharA0('S');
        uart_putcharA0('E');
        uart_putcharA0('T');
        uart_putcharA0('_');
    uart_putcharA0('C');
    uart_putcharA0('E');
    uart_putcharA0('N');
    uart_putcharA0('T');
    uart_putcharA0('=');
    uart_putcharA0('O');
    uart_putcharA0('N');
    uart_putcharA0('\r');
```

```
}

void centralOff(){
        uart_putcharA0('S');
        uart_putcharA0('E');
        uart_putcharA0('T');
        uart_putcharA0('_');
    uart_putcharA0('C');
    uart_putcharA0('E');
    uart_putcharA0('N');
    uart_putcharA0('T');
    uart_putcharA0('=');
    uart_putcharA0('O');
    uart_putcharA0('F');
    uart_putcharA0('F');
    uart_putcharA0('\r');

}

void reset(){
    uart_putcharA0('R');
    uart_putcharA0('S');
    uart_putcharA0('T');
    uart_putcharA0('\r');
}

void connect(){
        uart_putcharA0('C');
        uart_putcharA0('O');
        uart_putcharA0('N');
        uart_putcharA0('_');
    uart_putcharA0('C');
    uart_putcharA0('1');
    uart_putcharA0('B');
    uart_putcharA0('C');
    uart_putcharA0('9');
    uart_putcharA0('C');
    uart_putcharA0('9');
    uart_putcharA0('B');
    uart_putcharA0('5');
    uart_putcharA0('7');
    uart_putcharA0('D');
    uart_putcharA0('6');
    uart_putcharA0('_');
    uart_putcharA0('0');
    uart_putcharA0('\r');
}

void cconOff(void){
        uart_putcharA0('S');
        uart_putcharA0('E');
        uart_putcharA0('T');
        uart_putcharA0('_');
    uart_putcharA0('C');
    uart_putcharA0('C');
```

```
    uart_putcharA0('O');
    uart_putcharA0('N');
    uart_putcharA0('=');
    uart_putcharA0('O');
    uart_putcharA0('F');
    uart_putcharA0('F');
    uart_putcharA0('\r');
}

void setScanInterval(){
        uart_putcharA0('S');
        uart_putcharA0('E');
        uart_putcharA0('T');
        uart_putcharA0('_');
    uart_putcharA0('S');
    uart_putcharA0('C');
    uart_putcharA0('N');
    uart_putcharA0('P');
    uart_putcharA0('=');
    uart_putcharA0('0');
    uart_putcharA0('0');
    uart_putcharA0('1');
    uart_putcharA0('F');
    uart_putcharA0('4');
    uart_putcharA0('0');
    uart_putcharA0('0');
    uart_putcharA0('0');
    uart_putcharA0('_');
    uart_putcharA0('0');
    uart_putcharA0('0');
    uart_putcharA0('0');
    uart_putcharA0('F');
    uart_putcharA0('A');
    uart_putcharA0('0');
    uart_putcharA0('0');
    uart_putcharA0('0');
    uart_putcharA0('\r');
}

void setScanTimeout(){
        uart_putcharA0('S');
        uart_putcharA0('E');
        uart_putcharA0('T');
        uart_putcharA0('_');
    uart_putcharA0('S');
    uart_putcharA0('C');
    uart_putcharA0('N');
    uart_putcharA0('T');
    uart_putcharA0('=');
    uart_putcharA0('0');
    uart_putcharA0('0');
    uart_putcharA0('0');
    uart_putcharA0('A');
    uart_putcharA0('\r');
}
```

```
void scanDog1(){
        uart_putcharA0('S');
        uart_putcharA0('C');
        uart_putcharA0('N');
        uart_putcharA0('␣');
    uart_putcharA0('B');
    uart_putcharA0('␣');
    uart_putcharA0('C');
    uart_putcharA0('1');
    uart_putcharA0('B');
    uart_putcharA0('C');
    uart_putcharA0('9');
    uart_putcharA0('C');
    uart_putcharA0('9');
    uart_putcharA0('B');
    uart_putcharA0('5');
    uart_putcharA0('7');
    uart_putcharA0('D');
    uart_putcharA0('6');
    uart_putcharA0('␣');
    uart_putcharA0('T');
    uart_putcharA0('␣');
        uart_putcharA0('0');
        uart_putcharA0('␣');
    uart_putcharA0('N');
    uart_putcharA0('␣');
        uart_putcharA0('B');
        uart_putcharA0('C');
    uart_putcharA0('0');
        uart_putcharA0('0');
        uart_putcharA0('0');
        uart_putcharA0('0');
    uart_putcharA0('1');
    uart_putcharA0('␣');
        uart_putcharA0('F');
        uart_putcharA0('␣');
        uart_putcharA0('6');
    uart_putcharA0('␣');
    uart_putcharA0('R');
    uart_putcharA0('␣');
        uart_putcharA0('E');
        uart_putcharA0('3');
    uart_putcharA0('␣');
        uart_putcharA0('M');
        uart_putcharA0('␣');
        uart_putcharA0('0');
        uart_putcharA0('\r');
}

void storeChanges(){
    uart_putcharA0('W');
        uart_putcharA0('R');
        uart_putcharA0('T');
        uart_putcharA0('\r');
}
void restore(){
```

```
    uart_putcharA0('R');
        uart_putcharA0('T');
        uart_putcharA0('R');
        uart_putcharA0('\r');
}
```

## 5.2 MSP Code: timer.h

```
/*
 * Clint Olsen and Matt Zarekani
 * ECEN 3360
 * Final Project: Automatic Dog Door
 *
 */
#ifndef TIMER_H_
#define TIMER_H_

void configure_timer(void);
void enable_timer_interrupts(void);
void disable_timer_interrupts(void);

#endif /* TIMER_H_ */
```

## 5.3 MSP Code: timer.c

```
/*
 * Clint Olsen and Matt Zarekani
 * ECEN 3360
 * Final Project: Automatic Dog Door
 *
 */

#include "timer.h"
#include "msp.h"
#include <stdint.h>


void configure_timer(void) {
        TA0CCR0 = 374; // Capture Compare Value (1 ms)
        //TA0CCTL0 = TIMER_A_CCTLN_CCIE; // TACCR0 interrupt enabled
        // up mode, SMCLK, enable
        TA0CTL = TIMER_A_CTL_MC__UP | TIMER_A_CTL_TASSEL_2  | TIMER_A_CTL_ID__8; //|  TIMER_A_CTL_IE
        TA0R = 0;
        /* NVIC Port Interrupt Enable Code */  // Enable interrupt in NVIC
        NVIC_EnableIRQ(TA0_0_IRQn);

}

void enable_timer_interrupts(void){
        TA0CCTL0 |= TIMER_A_CCTLN_CCIE; // TACCR0 interrupt enabled
}

void disable_timer_interrrupts(void){
        TA0CCTL0 &= ~TIMER_A_CCTLN_CCIE; // TACCR0 interrupt enabled
```

```
}
```

## 5.4 MSP Code: gpio.h

```
/*
 * Clint Olsen and Matt Zarekani
 * ECEN 3360
 * Final Project: Automatic Dog Door
 *
 */

#ifndef GPIO_H_
#define GPIO_H_

void configure_pins(void);

#endif /* GPIO_H_ */
```

## 5.5 MSP Code: gpio.c

```
/*
 * Clint Olsen and Matt Zarekani
 * ECEN 3360
 * Final Project: Automatic Dog Door
 *
 */
#include "gpio.h"
#include "msp.h"
#include <stdint.h>

void configure_pins(void) {
        /* Port configuration code here for the switches or LEDs */

        //Configure LED
                P2DIR |= BIT5;
                P2OUT &= ~BIT5;
                P3DIR |= BIT0;
                P3OUT &= ~BIT0;

}
```

## 5.6 Java Code: MainActivity.java

```
package heyooo.pawavenue;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
```

```
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;


import app.akexorcist.bluetotohspp.library.BluetoothSPP;
import app.akexorcist.bluetotohspp.library.BluetoothState;
import app.akexorcist.bluetotohspp.library.DeviceList;

import static android.os.SystemClock.sleep;



public class MainActivity extends AppCompatActivity {
    private static final String TAG = MainActivity.class.getSimpleName();
    private Button smartLockConfig, smartLock, statusButton;

    public static BluetoothSPP bt;
    //private final InputStream InStream;
    TextView textStatus, smartLockStatus;
    ImageView directionImg;
    int position1 = 0; //smartLock spinner information, 0 = blocked, 1 = unblocked
    int position2 = 0;
    public static boolean connected = false;
    boolean smartLockBool = false;
    Menu menu;
    int dog1Status, dog2Status; //0 = indoors, 1 = outdoors
    @Override
    protected void onCreate(final Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        addListenerOnButton();
        if(getSharedPreferences("spinnerData", MODE_PRIVATE) != null) {
            SharedPreferences savedData = getSharedPreferences("spinnerData", MODE_PRIVATE);
            position1 = savedData.getInt("position1", 0);
            position2 = savedData.getInt("position2", 0);
        }
        Log.i("Check", "onCreate");
        directionImg = (ImageView)findViewById(R.id.imageView);
        directionImg.setImageResource(R.drawable.gate_open);

        textStatus = (TextView) findViewById(R.id.textStatus);
        smartLockStatus = (TextView) findViewById(R.id.smartLockStatus);

        bt = new BluetoothSPP(this);

        //if bluetooth is not available, return.
        if (!bt.isBluetoothAvailable()) {
            Toast.makeText(getApplicationContext()
                    , "Bluetooth is not available"
                    , Toast.LENGTH_SHORT).show();
            finish();
```

```
}


// display received data
bt.setOnDataReceivedListener(new BluetoothSPP.OnDataReceivedListener() {
    public void onDataReceived(byte[] data, String message) {
        SharedPreferences saveData = getSharedPreferences("statusData", MODE_PRIVATE);
        SharedPreferences.Editor edit = saveData.edit(); //save dog location data
        edit.commit();
        Log.d(TAG, message);
        if (message.substring(0,1).equals("U")) {
            Log.d(TAG, "unlocked");
            directionImg.setImageResource(R.drawable.gate_open);
            //textRead.append(message + "\n");
        } else if (message.substring(0,1).equals("L")) {
            directionImg.setImageResource(R.drawable.gate_closed);
        } else if (message.substring(0,1).equals("e")) {
            textStatus.setText("Already Locked");
        } else if (message.substring(0,1).equals("E")) {
            textStatus.setText("Already UnLocked");
        } else if (message.equals("1I")) {
            Log.i("Received","1. In doors");
            dog1Status = 0;
            edit.putInt("dog1Status", dog1Status);
            edit.commit();
        } else if (message.equals("1O")) {
            Log.i("Received","1. Outdoors");
            dog1Status = 1;
            edit.putInt("dog1Status", dog1Status);
            edit.commit();
        } else if (message.equals("2I")) {
            Log.i("Received","2. In doors");
            dog2Status = 0;
            edit.putInt("dog2Status", dog2Status);
            edit.commit();
        } else if (message.equals("2O")) {
            Log.i("Received","2. Outdoors");
            dog2Status = 1;
            edit.putInt("dog2Status", dog2Status);
            edit.commit();
        }
    }
});

//change the status label and menu options when connection status changed.
bt.setBluetoothConnectionListener(new BluetoothSPP.BluetoothConnectionListener() {
    public void onDeviceDisconnected() {
        textStatus.setText("Status : Not connect");
        menu.clear();
        getMenuInflater().inflate(R.menu.menu_connection, menu);
        connected = false;
    }

    public void onDeviceConnectionFailed() {
        textStatus.setText("Status : Connection failed");
    }
```

```
        public void onDeviceConnected(String name, String address) {
            connected = true;
            textStatus.setText("Status : Connected to " + name);
            menu.clear();
            getMenuInflater().inflate(R.menu.menu_disconnection, menu);
            sleep(100);
            SharedPreferences savedData = getSharedPreferences("spinnerData", MODE_PRIVATE);
            bt.send("C", true); //let door know we are connected
            sleep(100);
            if(savedData.getInt("position1", 0) == 0){ //matches configuration between app and door
                bt.send("1B", true); //Dog 1 blocked
            }else
                bt.send("1A", true); //Dog 1 allowed
            position1 = savedData.getInt("position1", 0);
            sleep(100);
            if(savedData.getInt("position2", 0) == 0){
                bt.send("2B", true); //Dog 2 blocked
            }else
                bt.send("2A", true); //Dog 2 allowed
            position2 = savedData.getInt("position2", 0);
        }
    });

}

//init option menu
public boolean onCreateOptionsMenu(Menu menu) {
    this.menu = menu;
    getMenuInflater().inflate(R.menu.menu_connection, menu);
    return true;
}

public void onDestroy() {
    Log.d(TAG, "close bt");
    super.onDestroy();
    bt.stopService();
}

//handle when an item in the menu options is selected.
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.menu_device_connect) {
        //connecting other kind of devices
        bt.setDeviceTarget(BluetoothState.DEVICE_OTHER);
        Intent intent = new Intent(getApplicationContext(), DeviceList.class);
        startActivityForResult(intent, BluetoothState.REQUEST_CONNECT_DEVICE);
    } else if (id == R.id.menu_disconnect) {
        //disconnect from a device
        if (bt.getServiceState() == BluetoothState.STATE_CONNECTED)
            bt.disconnect();
    }
    return super.onOptionsItemSelected(item);
}

public void onStart() {
```

```
        super.onStart();
        //enable bluetooth and bt service.
        if (!bt.isBluetoothEnabled()) {
            Intent intent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(intent, BluetoothState.REQUEST_ENABLE_BT);
        } else {
            if (!bt.isServiceAvailable()) {
                bt.setupService();
                bt.startService(BluetoothState.DEVICE_ANDROID);
                setup();
            }
        }
    }

    //setup button for sending data.
    public void setup() {
        Button lock = (Button) findViewById(R.id.lock);
        Button unlock = (Button) findViewById(R.id.unlock);
        lock.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Log.d(TAG, "sent");
                bt.send("L", true);
            }
        });
        unlock.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                bt.send("U", true);
            }
        });
    }

    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        switch(requestCode) {
            case (0) : {
                if (resultCode == Activity.RESULT_OK) {
                    String returnValue = data.getStringExtra("some_key");
                    Log.d(TAG, returnValue);
                    bt.send(returnValue, true);
                }
                break;
            }
        }
        if (requestCode == BluetoothState.REQUEST_CONNECT_DEVICE) {
            if (resultCode == Activity.RESULT_OK)
                bt.connect(data);
        } else if (requestCode == BluetoothState.REQUEST_ENABLE_BT) {
            if (resultCode == Activity.RESULT_OK) {
                bt.setupService();
                bt.startService(BluetoothState.DEVICE_ANDROID);
                setup();
            } else {
                Toast.makeText(getApplicationContext()
                        , "Bluetooth was not enabled."
                        , Toast.LENGTH_SHORT).show();
                finish();
```

```
            }
        }
    }
    public void addListenerOnButton(){
        smartLockConfig = (Button) findViewById(R.id.smartLockConfig);
        smartLock = (Button) findViewById(R.id.smartLock);
        statusButton = (Button) findViewById(R.id.statusbutton);
        smartLockConfig.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Intent myIntent = new Intent(v.getContext(), SmartLockConfig.class);
                startActivity(myIntent);
            }
        });
        smartLock.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                bt.send("S", true);
                if(connected == true){ //dont change modes unless connected to Bluetooth module
                    if(smartLockBool == false) {
                        smartLockStatus.setText("Mode: Smart Lock On");
                        smartLockBool = true;
                    }else{
                        smartLockStatus.setText("Mode: Smart Lock Off");
                        smartLockBool = false;
                    }
                }
            }
        });
        statusButton.setOnClickListener(new View.OnClickListener(){
            public void onClick(View v){
                Intent myIntent = new Intent(v.getContext(), StatusActivity.class);
                startActivity(myIntent);
            }
        });
    }
    @Override
    public void onResume(){
        SharedPreferences savedData = getSharedPreferences("spinnerData", MODE_PRIVATE); //load save data
        if(position1 != savedData.getInt("position1", 0)){
            if(savedData.getInt("position1", 0) == 0){
                bt.send("1B", true); //Dog 1 blocked
            }else
                bt.send("1A", true); //Dog 1 allowed
        }
        position1 = savedData.getInt("position1", 0);
        if(position2 != savedData.getInt("position2", 0)){
            if(savedData.getInt("position2", 0) == 0){
                bt.send("2B", true); //Dog 2 blocked
            }else
                bt.send("2A", true); //Dog 2 allowed
        }
        position2 = savedData.getInt("position2", 0);
        Log.i("main","resume");
        super.onResume();
    }
}
```

## 5.7 Java Code: StatusActivity.java

```
package heyooo.pawavenue;

import android.content.SharedPreferences;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
import static heyooo.pawavenue.MainActivity.bt; //allows us to send commands from outside MainActivity
import static heyooo.pawavenue.MainActivity.connected; //are we connected?

public class StatusActivity extends AppCompatActivity implements AdapterView.OnItemSelectedListener{
    ImageView dog1Status, dog2Status;
    private Button refresh, saveButton;
    TextView accessStatus1, accessStatus2;
    Spinner locationSpinner1, locationSpinner2;
    int position1, position2;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_status);
        position1 = 0;
        position2 = 0;
        dog1Status = (ImageView)findViewById(R.id.dog1Status);
        dog2Status = (ImageView)findViewById(R.id.dog2Status);
        locationSpinner1 = (Spinner) findViewById(R.id.locationSpinner1); //spinner creation
        locationSpinner2 = (Spinner) findViewById(R.id.locationSpinner2);
        ArrayAdapter<CharSequence> adapter1 = ArrayAdapter.createFromResource(this,
                R.array.location_array, android.R.layout.simple_spinner_item);
        adapter1.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        locationSpinner1.setAdapter(adapter1);
        ArrayAdapter<CharSequence> adapter2 = ArrayAdapter.createFromResource(this,
                R.array.location_array, android.R.layout.simple_spinner_item);
        adapter2.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        locationSpinner2.setAdapter(adapter2);
        locationSpinner1.setOnItemSelectedListener(this);
        locationSpinner2.setOnItemSelectedListener(this);



        addListenerOnButton();
        refreshSettings();
        Log.i("StatusActivity","Created");

    }
    public void onItemSelected(AdapterView<?> parent, View view,
                               int pos, long id) {
        // An item was selected
        String item = parent.getItemAtPosition(pos).toString();
```

```
        if(parent == locationSpinner1) {
            position1 = pos;
        }
        else if(parent == locationSpinner2){
            position2 = pos;
        }
        Toast.makeText(parent.getContext(), "Selected: " + item, Toast.LENGTH_LONG).show();
        // Showing selected spinner item


    }
    public void onNothingSelected(AdapterView<?> parent) { //necessary for OnItemSelectedListener
        // Another interface callback
    }
    public void addListenerOnButton() {
        refresh = (Button) findViewById(R.id.refreshButton);
        saveButton = (Button) findViewById(R.id.saveButton);
        refresh.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                refreshSettings();
                Toast.makeText(StatusActivity.this, "Refreshed", Toast.LENGTH_LONG).show();
            }
        });
        saveButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                if (connected == true) {
                    SharedPreferences savedStatusData = getSharedPreferences("statusData", MODE_PRIVATE);
                    SharedPreferences.Editor edit = savedStatusData.edit();
                    if (savedStatusData.getInt("dog1Status", 0) != position1) { //user overrides dog
                         position
                        Log.i("StatusActivity", "Override Position");
                        if (position1 == 0) {
                            dog1Status.setImageResource(R.drawable.puppy_indoors);
                            bt.send("1I", true);
                        } else {
                            dog1Status.setImageResource(R.drawable.puppy_outdoors);
                            bt.send("1O", true);
                        }
                        edit.putInt("dog1Status", position1);
                    }
                    if (savedStatusData.getInt("dog2Status", 0) != position2) { //user overrides dog
                         position
                        Log.i("StatusActivity", "Override Position");
                        if (position2 == 0) {
                            dog2Status.setImageResource(R.drawable.puppy_indoors);
                            bt.send("2I", true);
                        } else {
                            dog2Status.setImageResource(R.drawable.puppy_outdoors);
                            bt.send("2O", true);
                        }
                        edit.putInt("dog2Status", position2);

                    }
                    edit.commit(); //save
                    Toast.makeText(StatusActivity.this, "Saved", Toast.LENGTH_LONG).show();
                }else{
```

```
                    Toast.makeText(StatusActivity.this, "Not Connected. Failed to Save", Toast.LENGTH_LONG
                        ).show();
                }
            }
        });
    }
    public void refreshSettings(){ //loads data from file
        accessStatus1 = (TextView) findViewById(R.id.accessStatus1);
        accessStatus2 = (TextView) findViewById(R.id.accessStatus2);
        SharedPreferences savedStatusData = getSharedPreferences("statusData", MODE_PRIVATE);

        if (savedStatusData.getInt("dog1Status", 0) == 0){
            Log.i("StatusActivity","Dog 1 Indoors");
            position1 = 0;
            dog1Status.setImageResource(R.drawable.puppy_indoors);
            accessStatus1.setText("Inside");
        }else{
            Log.i("StatusActivity","Dog 1 Outdoors");
            position1 = 1;
            dog1Status.setImageResource(R.drawable.puppy_outdoors);
            accessStatus1.setText("Outside");
        }
        locationSpinner1.setSelection(position1);
        if (savedStatusData.getInt("dog2Status", 0) == 0){
            Log.i("StatusActivity","Dog 2 Indoors");
            position2 = 0;
            dog2Status.setImageResource(R.drawable.puppy_indoors);
            accessStatus2.setText("Inside");
        }else{
            Log.i("StatusActivity","Dog 2 Outdoors");
            position2 = 1;
            dog2Status.setImageResource(R.drawable.puppy_outdoors);
            accessStatus2.setText("Outside");
        }
        locationSpinner2.setSelection(position2);

    }
    public void onResume(){
        refreshSettings();
        Log.i("StatusActivity","Resume");
        super.onResume();
    }
}
```

## 5.8 Java Code: SmartLockConfig.java

```
package heyooo.pawavenue;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
```

```
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Spinner;
import android.widget.Toast;


public class SmartLockConfig extends AppCompatActivity implements AdapterView.OnItemSelectedListener {
    Spinner spinner1;
    Spinner spinner2;
    int position1;
    int position2;
    private Button saveButton;
    ImageView Dog1Img,Dog2Img;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.smart_lock_config);
        addListenerOnButton();

        Dog1Img = (ImageView)findViewById(R.id.Dog1Img);
        Dog2Img = (ImageView)findViewById(R.id.Dog2Img);

        spinner1 = (Spinner) findViewById(R.id.smart_spinner1);
        spinner2 = (Spinner) findViewById(R.id.smart_spinner2);

        // Create ArrayAdapter
        ArrayAdapter<CharSequence> adapter1 = ArrayAdapter.createFromResource(this,
                R.array.dogs_array, android.R.layout.simple_spinner_item);
        adapter1.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        spinner1.setAdapter(adapter1);
        ArrayAdapter<CharSequence> adapter2 = ArrayAdapter.createFromResource(this,
                R.array.dogs_array, android.R.layout.simple_spinner_item);
        adapter2.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        // Apply the adapter to the spinner
        spinner2.setAdapter(adapter2);
        spinner1.setOnItemSelectedListener(this);
        spinner2.setOnItemSelectedListener(this);

    }

    public void onItemSelected(AdapterView<?> parent, View view,
                               int pos, long id) {
        String item = parent.getItemAtPosition(pos).toString();


        // Showing selected spinner item
        if(parent == spinner1) {
            position1 = pos;
            if(position1 == 0){//blocked
                Dog1Img.setImageResource(R.drawable.clifford_sad);
            }else{
                Dog1Img.setImageResource(R.drawable.clifford_happy);
            }
        }
```

```
        else if(parent == spinner2){
            position2 = pos;
            if(position2 == 0){//blocked
                Dog2Img.setImageResource(R.drawable.clifford_sad);

            }else{
                Dog2Img.setImageResource(R.drawable.clifford_happy);
            }
        }
        Toast.makeText(parent.getContext(), "Selected: " + item, Toast.LENGTH_LONG).show();

    }
    public void onNothingSelected(AdapterView<?> parent) {
        // Another interface callback
    }
    @Override
    public void onRestoreInstanceState(Bundle savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);
        int myInt = savedInstanceState.getInt("Spinner1");
        spinner1.setSelection(myInt);
        // Restore UI state from the savedInstanceState.
    }
    @Override
    public void onResume (){ //load data
        Log.i("Resume","Position 1" + position1);
        SharedPreferences example = getSharedPreferences("spinnerData", MODE_PRIVATE);
        position1 = example.getInt("position1", 0);
        position2 = example.getInt("position2", 0);
        spinner1.setSelection(position1);
        spinner2.setSelection(position2);

        super.onResume();
    }
    public void addListenerOnButton(){
        saveButton = (Button) findViewById(R.id.save);
        saveButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) { //write data
                SharedPreferences saveData = getSharedPreferences("spinnerData", MODE_PRIVATE);
                SharedPreferences.Editor edit = saveData.edit();
                edit.putInt("position1", position1);
                edit.putInt("position2", position2);
                edit.commit();
                Intent myIntent = new Intent(v.getContext(), MainActivity.class);
                myIntent.setFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
                startActivityIfNeeded(myIntent, 0);
            }
        });
    }
}
```