ECE 110 Honors Project Report

Maddie Wilson
Mingze Xiao
Pranav Perepa

Introduction

Problem Statement

Our project aimed to create a wireless device that could be placed over a light switch and turn it on or off by using a phone or remote. This project would solve the problem of wanting to turn off the light from bed or across a room. An added feature would be to make it controllable from far away (like from another building) so you could check your phone to see if you had remembered to turn your light off and if you had, be able to switch it off remotely. The project will involve two components: the dial/mechanical component placed over the light switch and the phone app interface.

Basic Overview of Proposed Solution

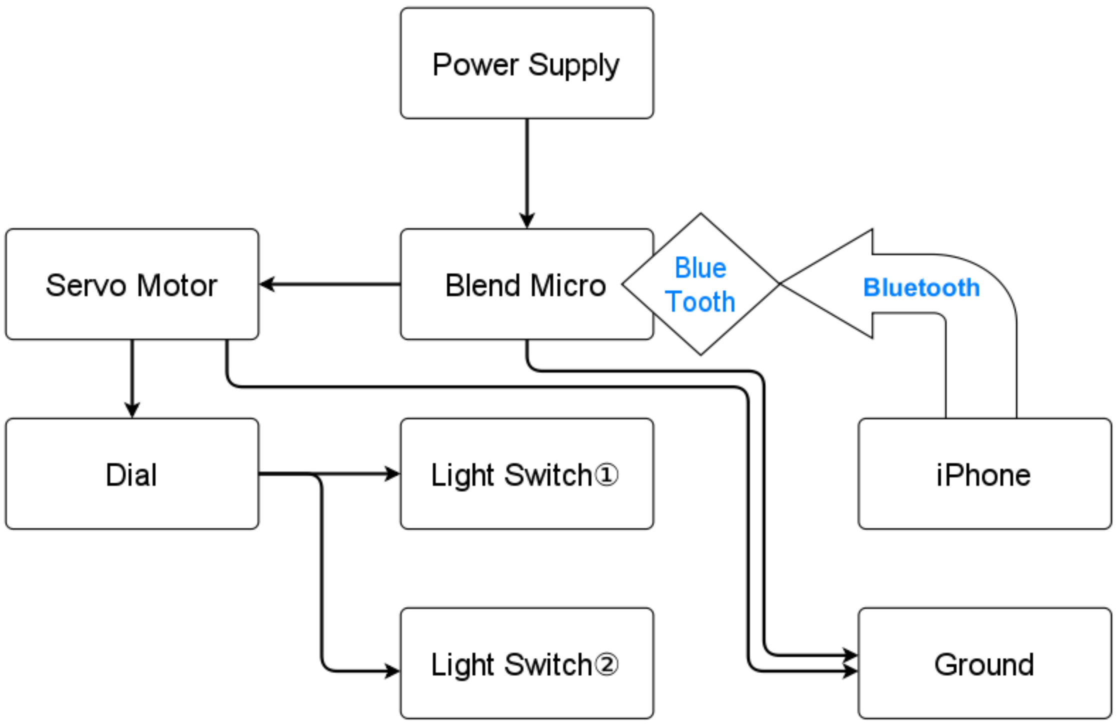
Our proposed solution was to design a circular dial with a hole towards one edge that the light switch toggle could be placed into. The dial be attached in the center to a servo motor. When the servo turned, it would shift the hole downwards causing the light switch to flip off. Turning the servo the other way would allow for the light switch to be flipped back on.

We also wanted to incorporate the ability to flip a paddle switch as well. To do this, we would add a bump to the other side of the dial. The bump would perform a similar function as the hole in the dial except press down on one side of the switch or the other as the servo turned the dial.

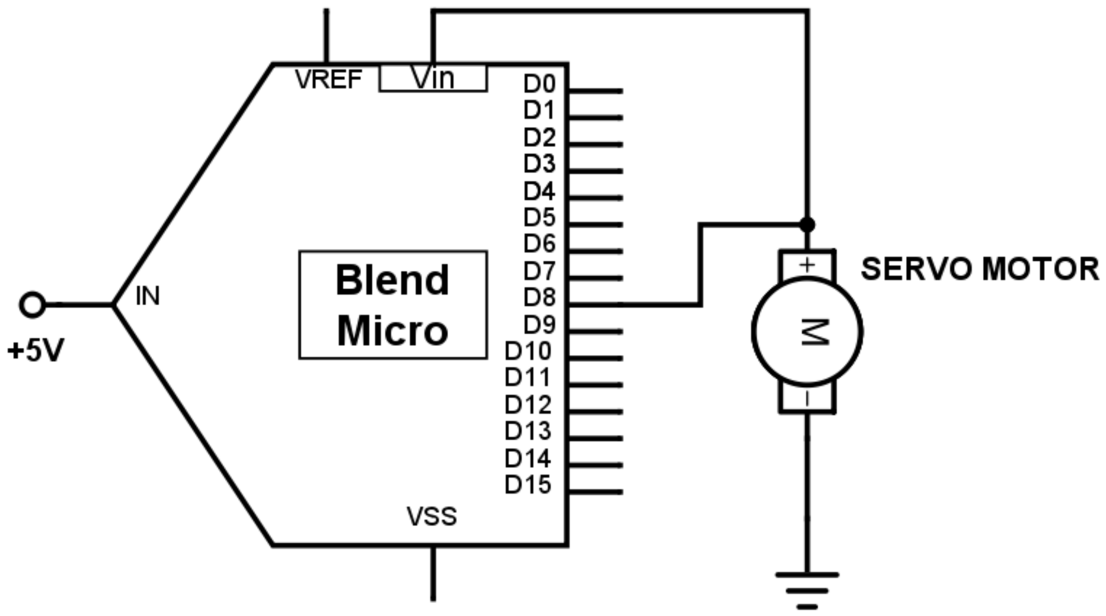
To control the servo, we wanted to build a user-friendly app interface. To do this we needed a bluetooth enabled Arduino Board. We ended up going with the Blend Micro board because it was a combined BLE and Arduino board and allowed us to connect with it from nearby and far away. During testing, our furthest test found that we were able to control the servo from behind a door about 20 feet away. On the UI would be a scroll bar By shifting a scroll bar, the user could turn the servo one way or another, flipping the light switch.

Design

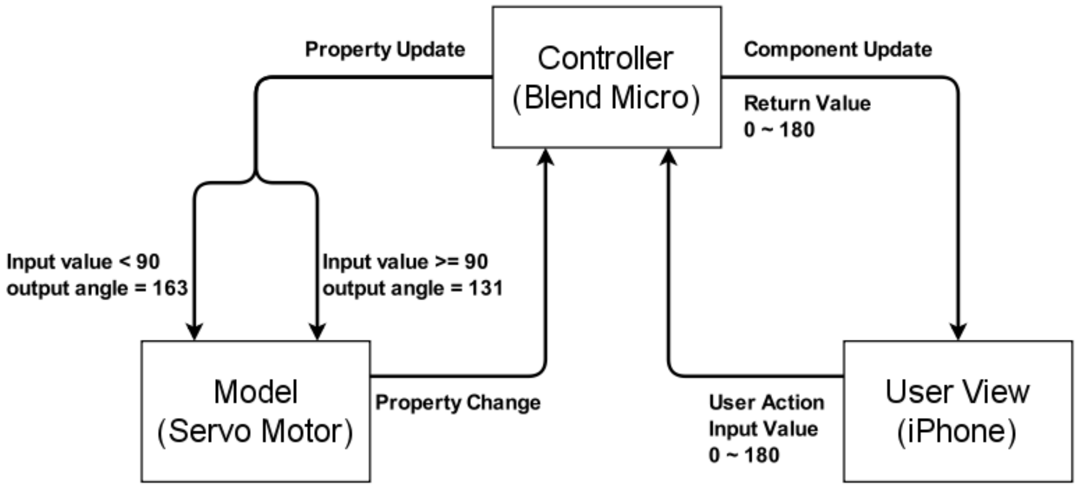
Block Diagram



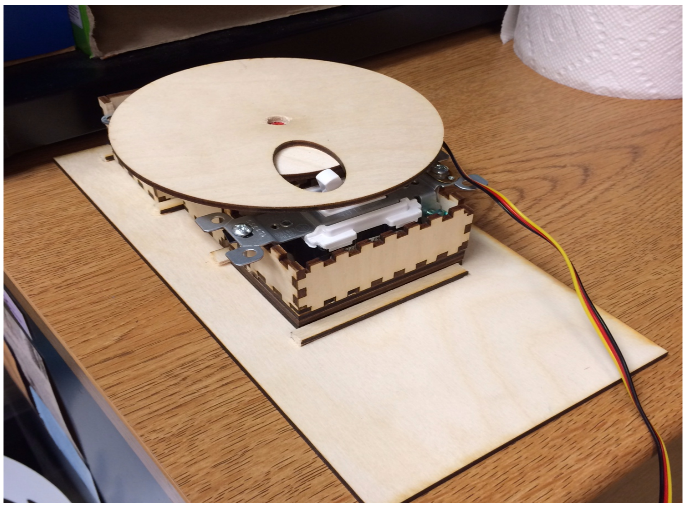
Circuit Schematics

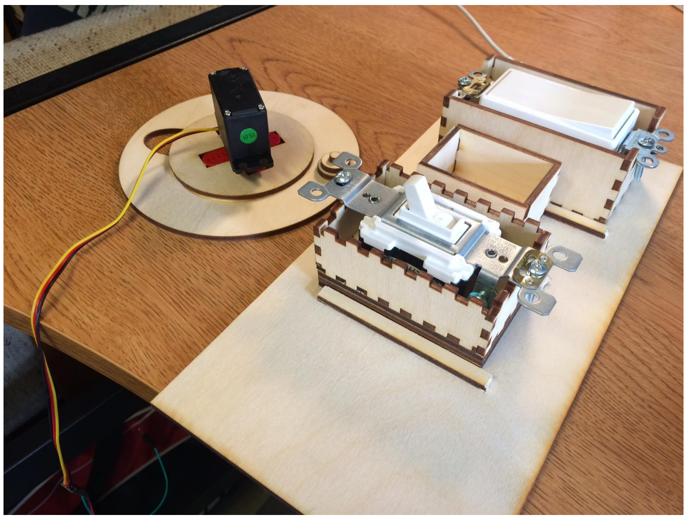


Flow Chart of Software



Pictures of Device





Code Appendix (Related Parts)

Blend Micro Code (Description)

The Blend Micro has an example code in order to teach us how to use the basic functions. We changed the servo portion of the example code to achieve our goal. In the example code, it has two files. One (child class) inherits from the other (father class). The father class can initialize the board information according to which board is connected to our computer. The content of it is out of the scope of our knowledge and this course's goal. Fortunately, we realized that we do not need to change it, because all the servo part is in the child class.

In the the original child class, the code can control the servo motor to rotate from 0 to 180 degrees according to the input value from 0 to 180 received from your iPhone. However, we only need two degrees to turn on and off the light switch. Through testing physically, we find the two degrees are 131° and 163°. Then we used the [if … else] to control the output value to let the servo motor can only rotate to these two degrees (The blue part in Code Appendix). Then we tested it, it worked very well to turn on and off the light switch. But sometimes it would rotate randomly and we didn't know why. Because the chance of this kind of error is very low, we did not care about it at first. When the error appeared again, we realized that we had to fix it.

At first, we thought it was something wrong in the father class. After inspecting the father class, we found there was nothing that could result in the random rotation. Finally, we found in the child class that, besides the codes controlling the behavior of the servo motor, there is another part of code initializing the servo motor. If we did not change it, it will initialize to 90 degrees automatically, which is out of the scope of our physical design. If we prevent it from rotating to 90 degrees, it will rotate randomly and not rotate as we told it from iPhone. In addition, we only need the servo to rotate from 131 to 163 degrees. But the default range is 0 – 180. It might be another dangerous reason for error. Therefore, we changed the rotating range to 131 – 163 degrees and initialized the setup degree to 163 degrees (The red part in Code Appendix). After doing so, it never went wrong.

Child class

```
#include <Servo.h>
#include <SPI.h>
#include <boards.h>
#include <RBL_nRF8001.h>
#include "Boards.h"

#define PROTOCOL_MAJOR_VERSION   0 //
#define PROTOCOL_MINOR_VERSION   0 //
#define PROTOCOL_BUGFIX_VERSION  2 // bugfix

#define PIN_CAPABILITY_NONE      0x00
#define PIN_CAPABILITY_DIGITAL   0x01
#define PIN_CAPABILITY_ANALOG    0x02
#define PIN_CAPABILITY_PWM       0x04
```

```
#define PIN_CAPABILITY_SERVO     0x08
#define PIN_CAPABILITY_I2C       0x10

// pin modes
//#define INPUT                0x00 // defined in wiring.h
//#define OUTPUT               0x01 // defined in wiring.h
#define ANALOG                 0x02 // analog pin in analogInput mode
#define PWM                    0x03 // digital pin in PWM output mode
#define SERVO                  0x04 // digital pin in Servo output mode

byte pin_mode[TOTAL_PINS];
byte pin_state[TOTAL_PINS];
byte pin_pwm[TOTAL_PINS];
byte pin_servo[TOTAL_PINS];

Servo servos[MAX_SERVOS];

void setup()
{
  Serial.begin(57600);
  Serial.println("BLE Arduino Slave");

  /* Default all to digital input */
  for (int pin = 0; pin < TOTAL_PINS; pin++)
  {
    // Set pin to input with internal pull up
    pinMode(pin, INPUT);
    digitalWrite(pin, HIGH);

    // Save pin mode and state
    pin_mode[pin] = INPUT;
    pin_state[pin] = LOW;
  }

  // Default pins set to 9 and 8 for REQN and RDYN
  // Set your REQN and RDYN here before ble_begin() if you need
  //ble_set_pins(3, 2);

  // Set your BLE Shield name here, max. length 10
  //ble_set_name("My Name");

  // Init. and start BLE library.
  ble_begin();
}

            else if (mode == SERVO)
            {
              if (IS_PIN_SERVO(pin))
              {
                pin_servo[pin] = 0;
                pin_mode[pin] = SERVO;
                if (!servos[PIN_TO_SERVO(pin)].attached())
                  servos[PIN_TO_SERVO(pin)].attach(PIN_TO_DIGITAL(pin), 1897, 2226); //limit
range
                  servos[PIN_TO_SERVO(pin)].write(163); //initialize the angle
              }
            }
          }

case 'O': // set Servo
        {
          byte pin = ble_read(); // The pin we'll use
          byte value = ble_read(); // The input value from iPhone
          byte input = 163; // The initial input value (angle)
          if (value <= 90) input = 163; // make the input only two options
          if (value > 90) input = 131; // open and close

          if (IS_PIN_SERVO(pin))
            servos[PIN_TO_SERVO(pin)].write(input); // change the angle
          pin_servo[pin] = value; // report to the iPhone
          reportPinServoData(pin); // report to the Serial Monitor
        }
        break;
```

Father class (Boards.h)

```
#ifndef Boards_h
#define Boards_h

#include <inttypes.h>

#if defined(ARDUINO) && ARDUINO >= 100
#include "Arduino.h"  // for digitalRead, digitalWrite, etc
#else
#include "WProgram.h"
#endif

#ifndef MAX_SERVOS
#define MAX_SERVOS 0
#endif



#ifndef digitalPinHasPWM
#define digitalPinHasPWM(p)     IS_PIN_DIGITAL(p)
#endif



// Blend Micro
#elif defined(BLEND_MICRO)
#define TOTAL_ANALOG_PINS       6
#define TOTAL_PINS              24 // 11 digital + 12 analog
#define VERSION_BLINK_PIN       13
#define IS_PIN_DIGITAL(p)       ( (p) >= 0 && (p) < 24 && !((p) == 4) && !((p) >= 6 && (p)
<= 7) && !((p) >=14 && (p) <= 17) )
#define IS_PIN_ANALOG(p)        ((p) >= 18 && (p) < 24)
#define IS_PIN_PWM(p)           ( (p) == 3 || (p) == 5 || (p) == 9 || (p) == 10 || (p) == 11 || (p) ==
13 )
#define IS_PIN_SERVO(p)         ( (p) >= 0 && (p) < MAX_SERVOS && !((p) == 4)
&& !((p) >= 6 && (p) <= 7) )
#define IS_PIN_I2C(p)           ((p) == 5 || (p) == 6)
#define PIN_TO_DIGITAL(p)       (p)
#define PIN_TO_ANALOG(p)        ((p)-18)
#define PIN_TO_PWM(p)           PIN_TO_DIGITAL(p)
#define PIN_TO_SERVO(p)         (p)

static inline unsigned char writePort(byte, byte, byte) __attribute__((always_inline, unused));
static inline unsigned char writePort(byte port, byte value, byte bitmask)
{
#if defined(ARDUINO_PINOUT_OPTIMIZE)
        if (port == 0) {
                bitmask = bitmask & 0xFC;  // do not touch Tx & Rx pins
                byte valD = value & bitmask;
```

```
		byte maskD = ~bitmask;
		cli();
		PORTD = (PORTD & maskD) | valD;
		sei();
	} else if (port == 1) {
		byte valB = (value & bitmask) & 0x3F;
		byte valC = (value & bitmask) >> 6;
		byte maskB = ~(bitmask & 0x3F);
		byte maskC = ~((bitmask & 0xC0) >> 6);
		cli();
		PORTB = (PORTB & maskB) | valB;
		PORTC = (PORTC & maskC) | valC;
		sei();
	} else if (port == 2) {
		bitmask = bitmask & 0x0F;
		byte valC = (value & bitmask) << 2;
		byte maskC = ~(bitmask << 2);
		cli();
		PORTC = (PORTC & maskC) | valC;
		sei();
	}
#else
	byte pin=port*8;
	if ((bitmask & 0x01)) digitalWrite(PIN_TO_DIGITAL(pin+0), (value & 0x01));
	if ((bitmask & 0x02)) digitalWrite(PIN_TO_DIGITAL(pin+1), (value & 0x02));
	if ((bitmask & 0x04)) digitalWrite(PIN_TO_DIGITAL(pin+2), (value & 0x04));
	if ((bitmask & 0x08)) digitalWrite(PIN_TO_DIGITAL(pin+3), (value & 0x08));
	if ((bitmask & 0x10)) digitalWrite(PIN_TO_DIGITAL(pin+4), (value & 0x10));
	if ((bitmask & 0x20)) digitalWrite(PIN_TO_DIGITAL(pin+5), (value & 0x20));
	if ((bitmask & 0x40)) digitalWrite(PIN_TO_DIGITAL(pin+6), (value & 0x40));
	if ((bitmask & 0x80)) digitalWrite(PIN_TO_DIGITAL(pin+7), (value & 0x80));
#endif
}




#ifndef TOTAL_PORTS
#define TOTAL_PORTS		((TOTAL_PINS + 7) / 8)
#endif


#endif
```

Results

Our project result was just about the same as we imagined it would be at the beginning of the semester. We were able to use a servo connected to a dial to turn the switches on or off. The Blend Micro also gave us the capacity to control the device from far away through a scroll bar on the app UI. The toggle light switch was easy enough to turn once we figured out the correct angles for the code. The paddle switch proved to be a little more tricky. The dial turned to the appropriate angle to flip the switch but didn't have enough weight pressing on the switch to flip it.

One unexpected error we encountered during the demo was the servo turning without being told to. The servo seemed to initialize itself to a certain angle outside of the light switch's range. This caused the dial to pull the light switch out of it's holder. To fix this, we needed to alter the code. The code was a little tricky to navigate as the given example code was extremely long to allow for multiple pin controls. However, after identifying the correct portion of code we would need to manipulate to change the degrees for the servo, we were able to turn the dial the desired amount to flip the switches (see *Blend Micro Code (Description)* for more information).

Our device also required mechanical construction. To hold the switches and servo together, we went to the CU Community Fab Lab and designed three boxes attached to a board that could hold the different components in place. The dials the servo came with were also not big enough to reach the switches so we had to design a new dial ourselves. This dial was circular with a hole on one side for the toggle switch. On the other side, we glued three circles underneath the dial that would press down on the paddle switch. The hardest part of the mechanical design was sizing. To get the angles of the hole and circles on the dial right took careful measurement. Another tricky part was designing the box layout. We wanted a box for each of the switches and the servo. However, we wanted to be able to remove components if necessary so we created raised edges that would prevent the boxes from slipping across the board as the dial pushed the switches. Our servo box was also slightly too large for the servo so we added some duct tape padding in the box to keep the servo from shifting around as it turned. This part of the design process was a challenge but ultimately fun and turned out surprisingly well.

Future Work

Although our project serves an important purpose with its current design, there are definitely some areas where we can improve. Some ideas that we have for the most immediate next steps for improving our project are designing a better casing for the switches, specializing our design to work efficiently for just one switch, and creating a functionality for easy switching between Bluetooth control and manual control. Either way, the ECE 110 Honors Section has motivated us to continue using our technical knowledge to explore new ideas and solve problems.

Next Steps for Our Project

There are many different routes that we can take in order to improve our project, but each idea would serve a more specific purpose. Our first idea was to improve on our preliminary casing design by trying to make the casing less bulky. If our idea to use Bluetooth to control light switches were to actually be turned into a product, we would need to be able to install a mechanism in existing setups for light switches. We found that our original casing worked well as a prototype, but did not seamlessly flip the switch without assistance. In order to solve this problem, we could have the servo connecting to a small loop of cheese wire or durable string so that the setup is less noticeable. If we were building the design into a smart home, the Blend Micro could be included in the light switch system from the very beginning.

Another potential advancement to our project could utilize other projects, like a near-field infrared detector, so that when your phone is a certain distance away from the light switch it turns on and as it gets farther away, it turns off. This option would still require us to build a new casing for our light switch and require a more complicated circuit with the ability to read infrared feedback instead of the Blend Micro Chip to turn the servo.

Conclusion

Working on this ECE Honors Project was a very valuable experience for us. We gained a deeper understanding of how to work with different circuits and logic boards, not just the SparkFun RedBoard. Furthermore, we witnessed firsthand how important the applications of physics were to our project and many other projects around us. Paying a visit to the Fab Lab and creating our own external casing also provided a unique touch to our project while giving us a chance to learn how to use the tools and software that engineers use to make their prototypes.

What Worked?

During our demonstration, we were able to get our code to execute properly and the servo was able to turn the mechanism we built in order to flip the switch. We ran into a few issues after the first time in our demonstration, which are detailed below in the "What Didn't Work" section. So ultimately, we were able to achieve our goal—but our project still has a lot of room for improvement. Our basic original idea was executed through the use of the Blend Micro Chip. Using the Red Bear Lab smart phone application and the Blend Micro Chip, we were able to solve the problem of using Bluetooth to control our circuit. Solving this aspect of our project greatly expedited the process. After buying the Blend Micro Chip and writing the code to control the digital pin (which controlled the servo), we were able to get our project to work the way that we wanted it to.

We were surprised at the impressive range between the Red Bear Lab application and the Blend Micro, since Mingze was able to stand in another room about 10 feet away and still control the Bluetooth light switch. This capability kept us optimistic and interested in our project, and we plan to find new ways to take advantage of this powerful technology to perfect our Bluetooth switch design.

What Didn't Work?

Although we believed that our code would execute without any errors, we still faced some issues during our demonstration. When we connected the servo to our portable battery pack, our code did not work properly, since the servo was turning the plate at an angle that was too large—nearly rupturing our wooden design. Initially, we weren't sure if this was just caused by fluctuations in the voltage source (battery pack) or maybe it wasn't charged fully. However, after later stepping through the code and thoroughly debugging, we found that our code did not address certain special cases and conditions.

During the physical creation of our casing, we ran into problems in the early stages when we tried to use our SparkFun Kit servo motor to turn a switch, but found that it did not supply enough torque to turn the switch for the duration of the necessary angle. After crunching some numbers and determining the actual parameters that we needed a servo to have, we found that the ECE Supply Store offered another servo motor that could get the job done. Another problem that we had was that the circuit was not receiving enough power from the voltage source and through the breadboard. We fixed this problem by soldering the pins of the Blend Micro. We believe that even though the pins were soldered, certain weak connections to the Blend Micro contributed to the problems that we saw in our demonstration.

What Did You Learn?

From our demonstration, we learned the important lesson that even though you may think something will work properly, there are always going to be unforeseen factors that you cannot always control. After observing the sporadic behavior of the servo when connected to the voltage source, we were able to pinpoint possible sources of error. After tweaking the code and testing out different voltage source values, we were able to determine that the error was due to a combination of both. In the week after our demonstration we were able to revise our code as well as connect a steady voltage source and get the servo to flip the switch properly.

The entire Honors Project process, from start to finish, provided us with a fun challenge and an avenue to explore our creative interests. Each of us gained experience on what it feels like to be a part of the engineering design process, and we plan to continue to develop our Remote Switch project as well as pursue new ones.