5-2018

Multipurpose Iron Man Glove & Moveable Platform

Destinee Davis
Trinity University, ddavis2@trinity.edu

João Marques
Trinity University, jmarques@trinity.edu

Thomas Plantin
Trinity University, tplantin@trinity.edu

Multipurpose Iron Man Glove & Moveable Platform

Group D

Destinee Davis, João Marques, Thomas Plantin

(pledged)

April 30, 2018

Table of Contents

DESIGN SUMMARY:

The multipurpose glove is a hand-worn device that wirelessly controls the position of a platform and flashes a high power LED at a particular frequency in order to alias the motion of an oscillatory object. The position and motion of the platform is dictated by an accelerometer which is attached to the top of the glove, as shown in Fig. 1. The user can then tilt the hand forward, backward, left, or right, to cause the platform to move towards a desired location. The user can also adjust the flashing frequency of the high power LED that is attached to the bottom of the glove. This frequency adjustment can be done through hand motion and the accelerometer, or by changing the position of a potentiometer that is attached to the overall casing, as depicted by Fig. 1. Figure 1 also displays the battery armature, which is used to fix the battery and secure the system casing around the forearm of the user with VELCRO straps.

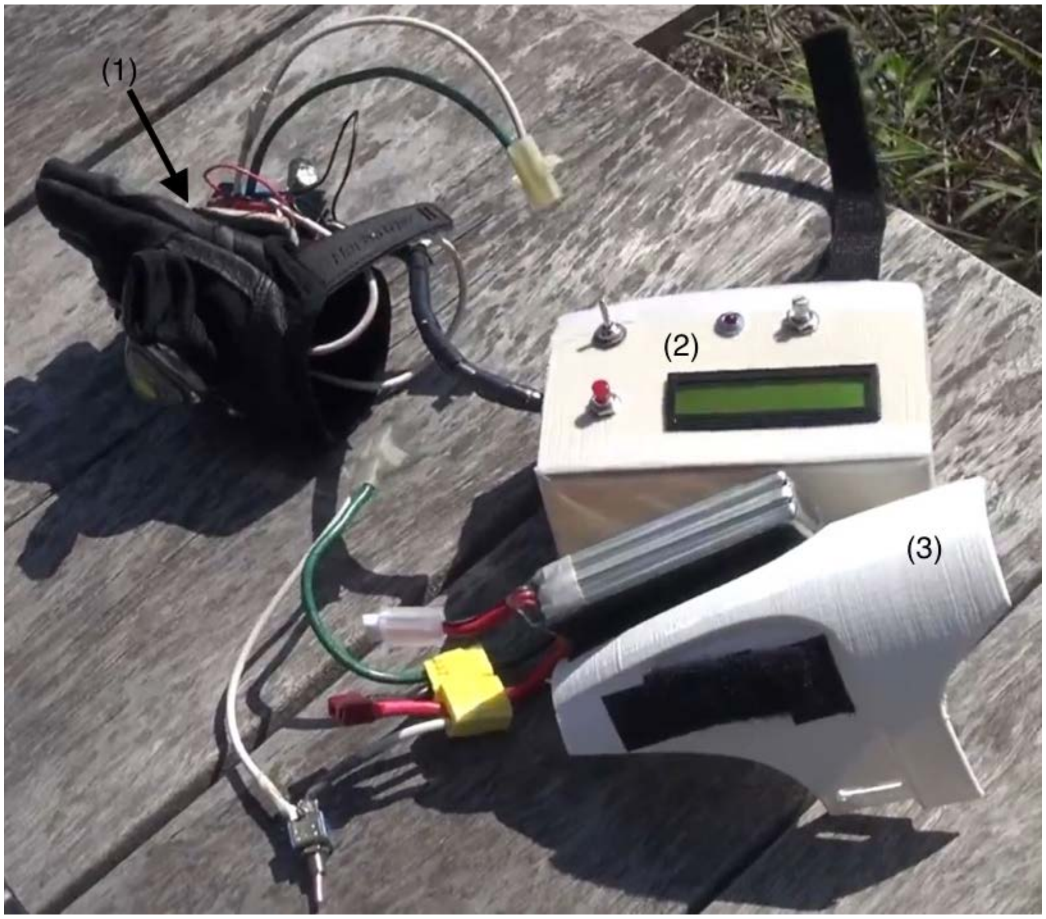


Figure 1: Final design prototype with the glove (1), the system casing (2), and the battery armature (3)

System Details:

Figure 2 exhibits a detailed picture of the design prototype with all the important user components. At first, the system is off, so to turn it on, the user must activate the main power switch. Once the switch is activated, power is delivered to the microcontrollers and electronics stored in the main casing. Battery power is also supplied to the DC-DC boost converter, which steps up the voltage from 11.1-V to 34-V in order for the high power LED to function when instructed.

The user now has a choice of three system modes, which are printed on the LCD. The first mode is the "Platform Control" mode, which allows the user to wirelessly control the platform's motion through the accelerometer. Also, if the fan switch is activated, then the fan on the platform will engage. The second mode is the "LED Hand Control" mode, which can be entered by pressing the red push button. On the button press, the user LED on the system casing will blink, the piezo buzzer will sound, and the LCD will update its display. In this second mode, the user can activate the high power LED by positioning his palm normal to the ground, and can adjust the frequency by tilting his hand left or right. Retracting the wrist downwards causes the high power LED to disengage. Finally, the third mode is the "LED Pot Control" mode, which is also accessed via the red push button. In this mode, the high power LED is automatically turned on and its flashing frequency can be altered with the potentiometer.

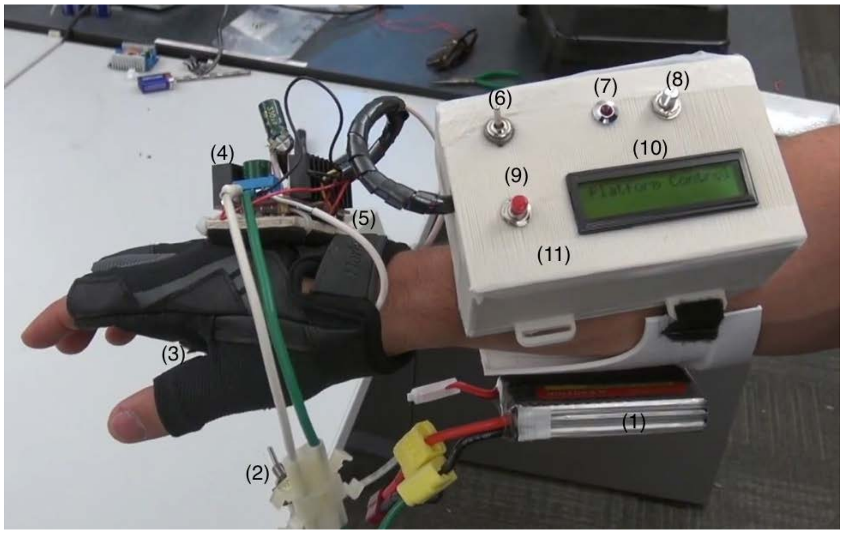


Figure 2: Detailed design prototype with the LiPo battery (1), the power switch (2), the high power LED (3), the DC-DC boost converter (4), the accelerometer (5), the fan switch (6), the user LED (7), the potentiometer (8), the push button (9), the LCD (10), and the piezo buzzer (11, within the main casing)

Figures A1-A4 in the appendix represent the functional diagrams of each microcontroller used in the project. Figure A1 combines the logic of two PIC microcontrollers, one which updates the display of the LCD, and one that both intakes the potentiometer value and communicates in binary with the Master Arduino. Portrayed in Fig. A2, the Master Arduino then reads the accelerometer value and outputs data to the transmitter. The transmitter then communicates the data to the receiver which then relays it to the Slave Arduino in Fig. A3. In essence, the Slave Arduino is merely a data bridge between the receiver and the platform PIC; it communicates in binary with the platform PIC, which then controls the motors and the fan according to Fig. A4.

Figures A5-A8 represent the various software flowchart. Figure A5 explains how the user interface scheme is structured, and Fig. A6-A8 explain the three principal modes that the system can enter while operating. From a more programming aspect, Fig. A9-A13 account for all of the necessary codes written both in the PIC Basic Pro and in the Arduino IDEs.

Finally, Fig. A14 displays the overall circuit schematic. This schematic is analyzed beginning from the lower-left corner where the Master PIC is positioned, and going around clockwise until the Platform PIC is reached. Reading the circuit this way enables one to follow what is occurring electrically among the microcontrollers, from the user input to the output signals linked to the actuators.

<u>DESIGN EVALUATION:</u>

The device successfully meets each of the requirements listed in the functional element categories. Our design meets the *Output Display* requirement by implementing a user LED as well as an LCD which prints the mode that the system is in. The *Audio Output Device* requirement is also fulfilled with the use of a piezo buzzer, which alerts the user when the main push button is pressed. The *Manual User Input* requirement is met with the use of a switch for the fan, of a push button for the system mode, and of a potentiometer for the adjustment of the flashing frequency of the high power LED. The requirement for the *Automatic Sensor* is satisfied with the use of an accelerometer, which both controls the flashing frequency of the high power LED and the motion of the platform. Our design achieves the *Actuators, Mechanisms & Hardware* section by implementing a servo motor controlled with a PWM signal. In addition, for this section, some 3D-printed parts were generated for the system casing and for the battery armature. Finally, for the *Logic, Processing, and Control* requirement, our device uses calculations (to output the correct flashing frequency of the high power LED to alias an oscillatory mechanism), multiple interfaced microcontrollers (interfacing the PICs with the Arduinos with binary communication), and an RF transmitter and receiver as components not included in other categories.

PARTIAL PARTS LIST:

Part Name/Description	Model Number	Source (vendor)	Price
DC-DC High Power Boost Converter	Hyelesiontek DC-DC Boost Converter	Ali Express	$3.45
2.4GHz RF Transmitter and Receiver Module	NRF24L01+	Amazon	$2.40
Accelerometer	MPU-6050	Banggood	$2.66
Arduino Nano (2)	Elegoo Nano board CH340	Amazon	$8.57

LESSONS LEARNED:

Throughout the design and implementation of the device, our group encountered many problems and difficulties. The first main difficulty that we faced was how to setup a PIC microcontroller both to output to an LCD and intake the value of a potentiometer. In order to enable the PIC to output user information to an LCD, all of the PIC's A/D converters had to be turned off. However, we needed a PIC A/D converter to read the potentiometer's position. We solved this problem by interfacing two different PICs. These PICs communicated in binary via their I/O ports. Since the system only has three different modes, we were able to achieve full communication between both PICs with only two I/O pins on each PIC (for more detailed information, see Fig. A9 & A10).

Another difficulty that arose was the wireless communication between the hand module and the platform. Due to scarce information about PIC microcontrollers on the internet, we had a hard time achieving wireless communication between two PICs. However, we found a lot of information for wireless communication with Arduino systems, so we ended up using two Arduino Nanos, which made the setup and wireless communication tremendously easier.

One final problem that emerged during the testing of our product came from the DC-DC boost converter. One of the capacitors of the module failed, and we were not able to find another 1000-uF in time for the project demo the next morning. However, while searching for that component, we came across a box of 330-uF capacitors. We thus soldered three of them in parallel in a rosette-like shape, and replaced the 1000-uF capacitor with what we had just contrived. Basic circuit knowledge was a big help!

As far as recommendations go, we believe that the most important aspect of the project is to stay on schedule. Our group managed to advance at a regular pace throughout the semester and stay on schedule. If we had not done that, the problems we faced would have caused us to turn in our project late. On a more technical side, we would suggest to comment the code(s) properly and clearly. It helps everyone in the group to understand what is going on when the code(s) need(s) to be read. Finally, it is critical to update the circuit schematic as the circuit building takes place. It made it a lot easier to have an updated schematic when the circuit had to be transferred from the breadboard to the soldering board.

References:

David G. Alciatore, Michael B. Histand, 2012, “Introduction to Mechatronics and Measurement Systems”, McGraw Hill, 4th edition, pp 301.

“LoRa Module VS nRF24 VS Generic RF Module || Range & Power Test”, https://www.youtube.com/watch?v=nP6YuwNVoPU

Gustavo Murta, “Arduino Strobe Light”, https://github.com/Gustavomurta/Arduino-Strobe-Light/blob/master/Arduino_Strobe_Gustavo.ino

APPENDIX:

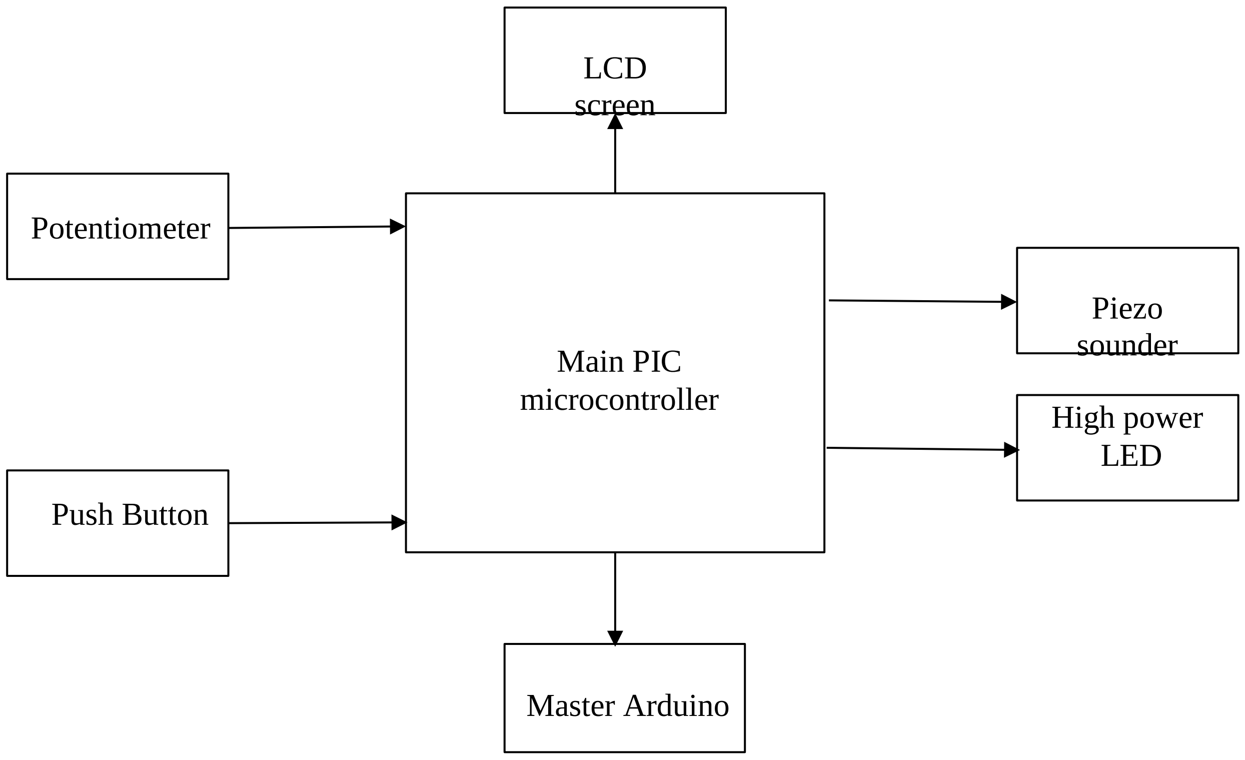


Figure A1: Functional diagram of the Main PIC microcontroller

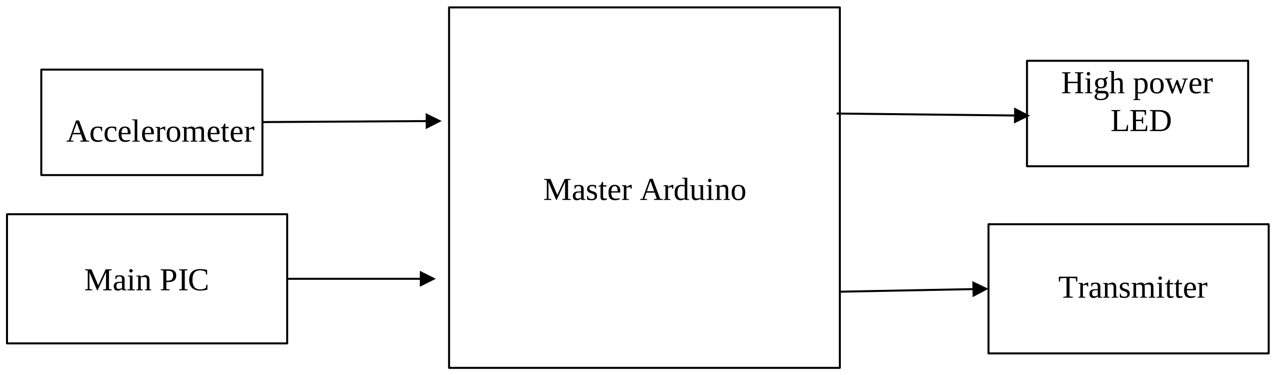


Figure A2: Functional diagram of the Master Arduino microcontroller

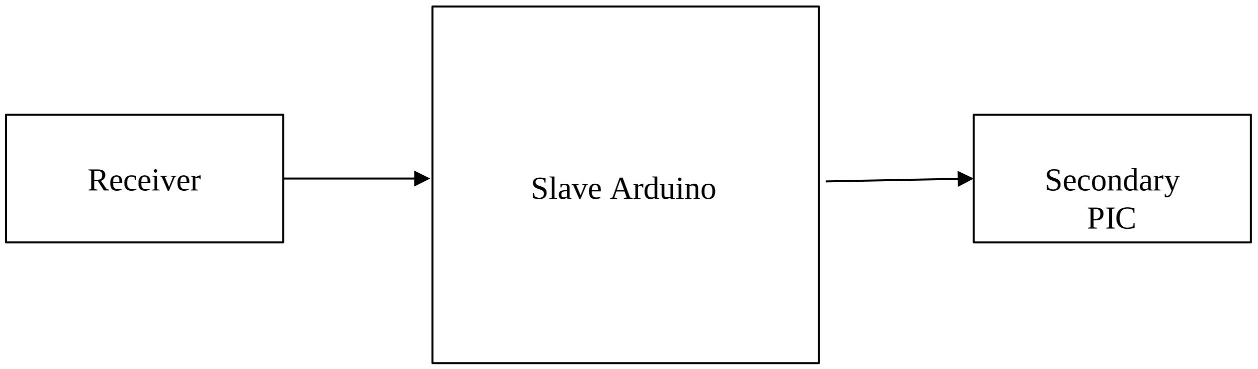


Figure A3: Functional diagram of the Slave Arduino microcontroller

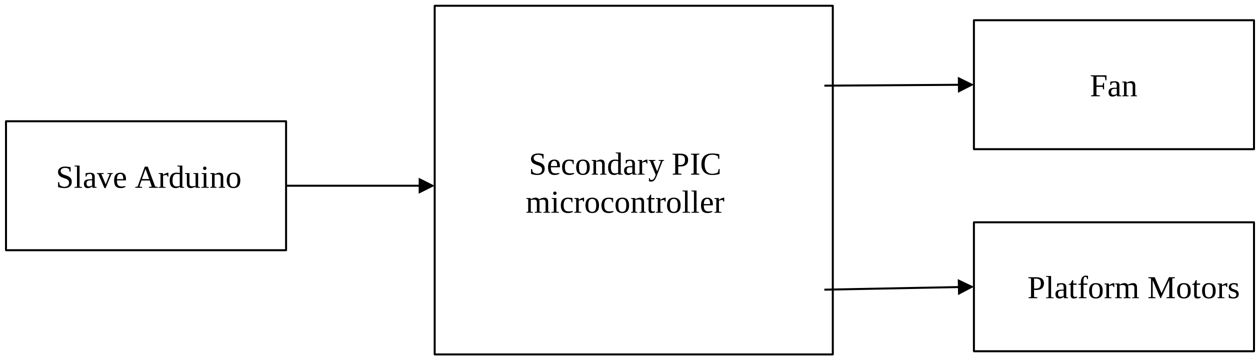


Figure A4: Functional diagram of the Secondary PIC microcontroller (platform PIC)

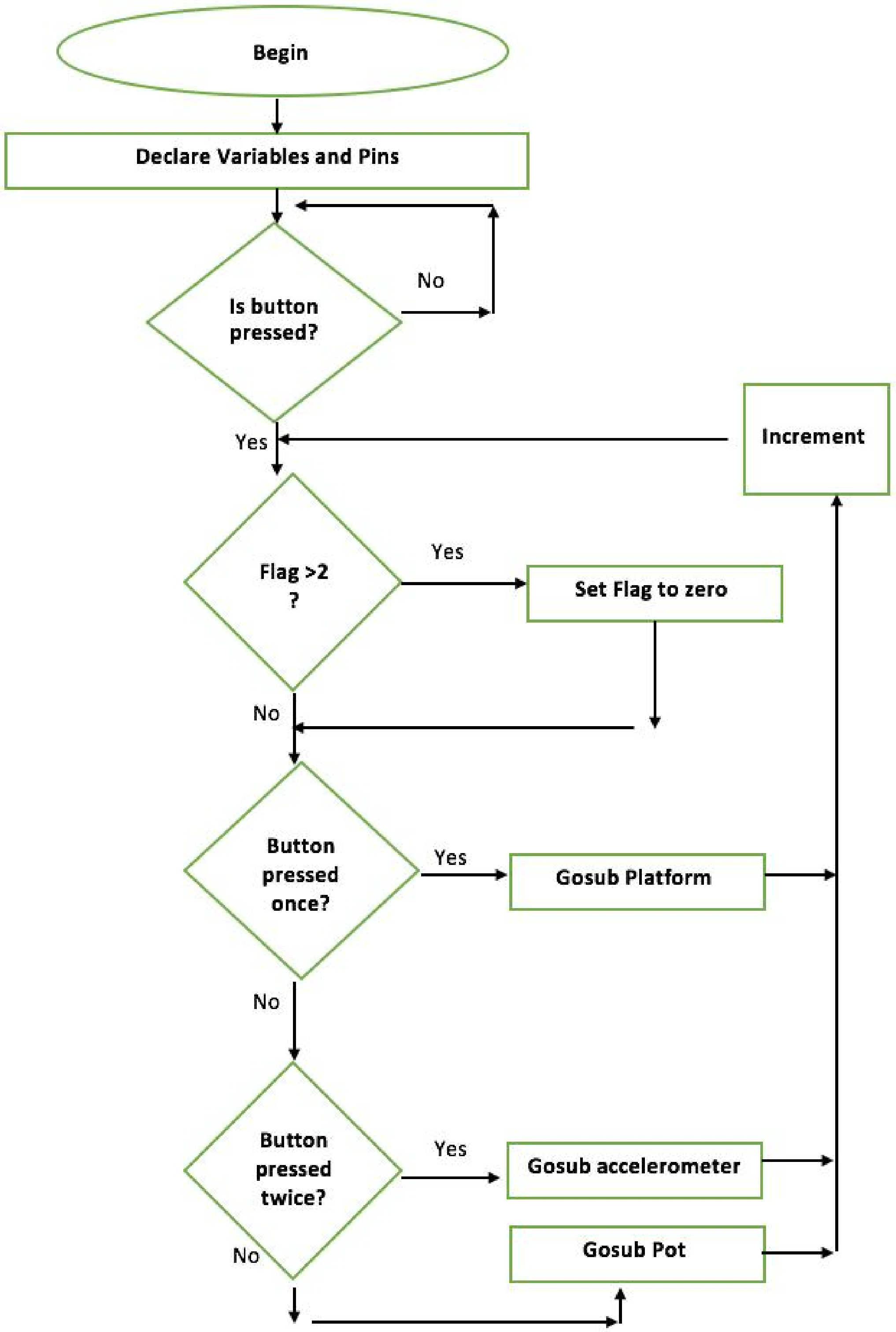


Figure A5: Software flowchart of the user interface

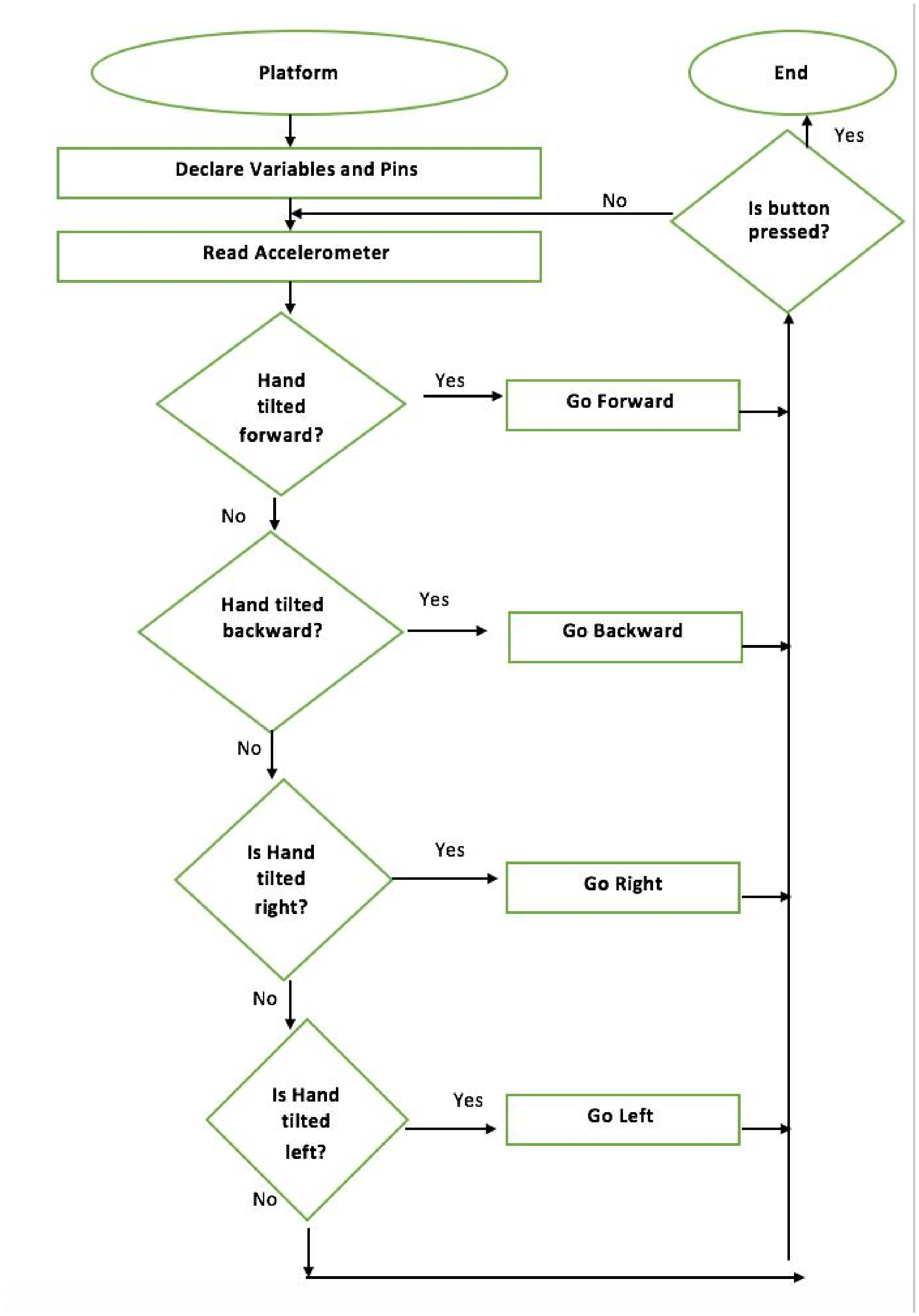


Figure A6: Software flowchart of the platform control

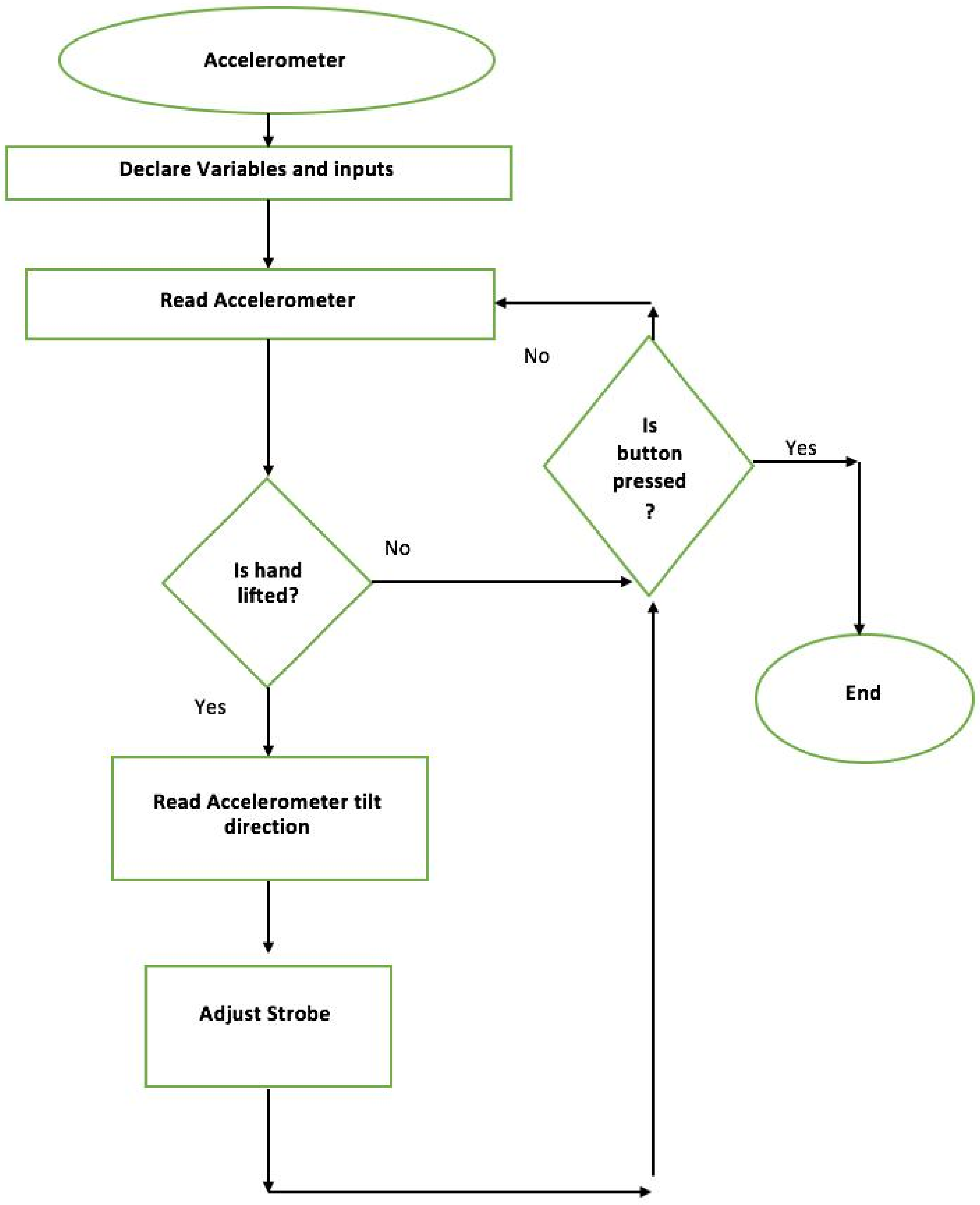


Figure A7: Software flowchart of the hand controlled LED flashing

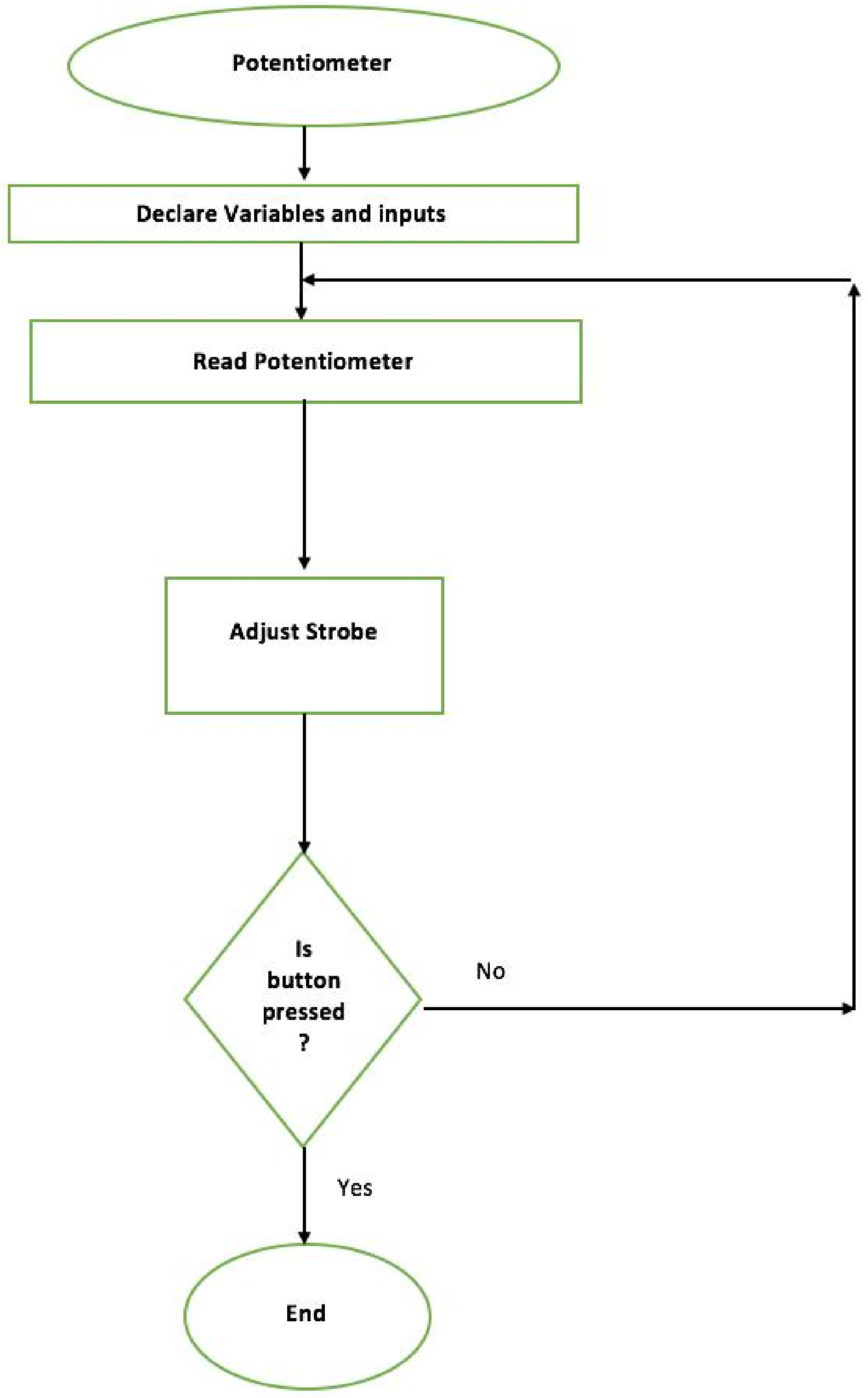
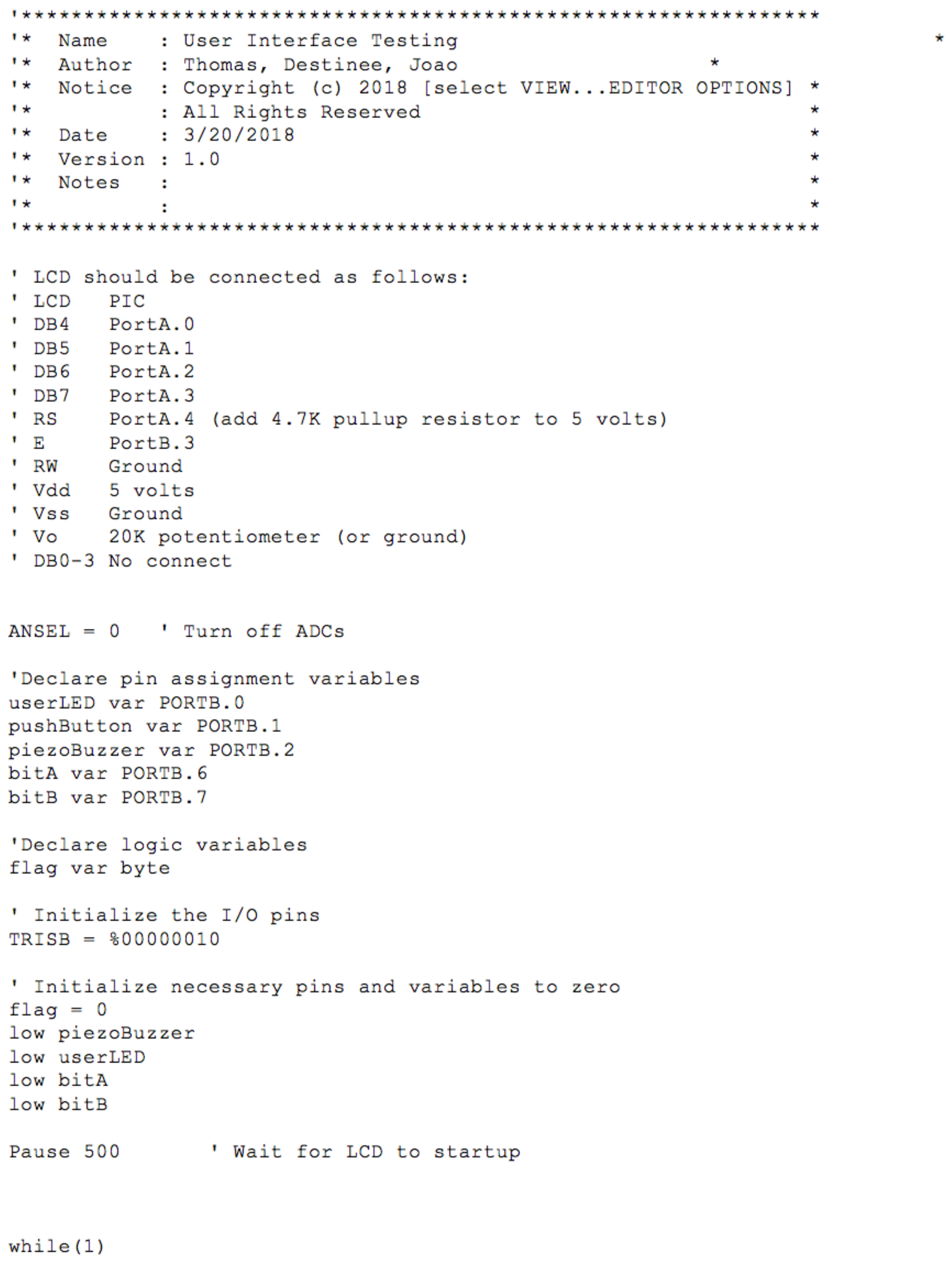


Figure A8: Software flowchart of the potentiometer controlled LED flashing



```
'****************************************************************
'*  Name    : User Interface Testing                                      *
'*  Author  : Thomas, Destinee, Joao                  *
'*  Notice  : Copyright (c) 2018 [select VIEW...EDITOR OPTIONS] *
'*          : All Rights Reserved                               *
'*  Date    : 3/20/2018                                         *
'*  Version : 1.0                                               *
'*  Notes   :                                                   *
'*          :                                                   *
'****************************************************************

' LCD should be connected as follows:
' LCD   PIC
' DB4   PortA.0
' DB5   PortA.1
' DB6   PortA.2
' DB7   PortA.3
' RS    PortA.4 (add 4.7K pullup resistor to 5 volts)
' E     PortB.3
' RW    Ground
' Vdd   5 volts
' Vss   Ground
' Vo    20K potentiometer (or ground)
' DB0-3 No connect


ANSEL = 0   ' Turn off ADCs

'Declare pin assignment variables
userLED var PORTB.0
pushButton var PORTB.1
piezoBuzzer var PORTB.2
bitA var PORTB.6
bitB var PORTB.7

'Declare logic variables
flag var byte

' Initialize the I/O pins
TRISB = %00000010

' Initialize necessary pins and variables to zero
flag = 0
low piezoBuzzer
low userLED
low bitA
low bitB

Pause 500       ' Wait for LCD to startup


while(1)
```

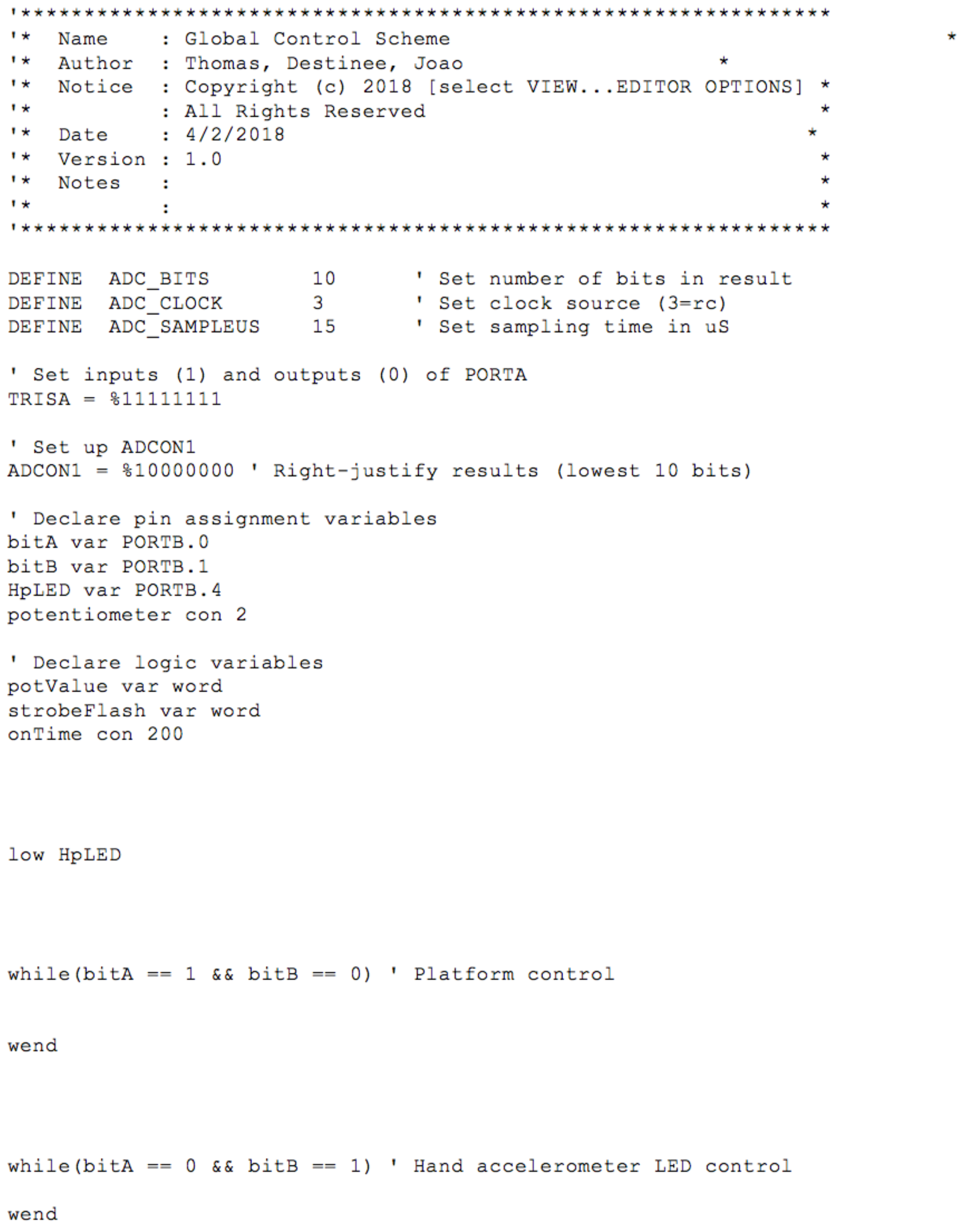
Figure A9: PIC Basic code of the user interface

```
   if (pushButton == 0) then ' Detect if button is pressed

        high piezoBuzzer  ' Warn user that button has been pressed
        high userLED

        if (flag > 2) then ' If button has been pressed more than 3 times, set
flag to zero
            flag = 0
        endif

        if(flag == 0) then ' First time pressed? Go to platform sub Routine
            pause 100
            low piezoBuzzer
            low userLED
            gosub platform
        endif

        if (flag == 1) then ' Second time pressed? Go to accel_LED sub Routine
            pause 100
            low piezoBuzzer
            low userLED
            gosub accel_LED
        endif

        if (flag == 2) then  ' Third time pressed? Go to pot_LED sub Routine
            pause 100
            low piezoBuzzer
            low userLED
            gosub pot_LED
        endif

        flag = flag + 1 ' Increment the flag



   endif



wend


platform:

    while(pushButton == 0) ' Wait until button is released to proceed
    wend

    Lcdout $fe, 1   ' Clear LCD screen
    Lcdout "Platform Control"  ' Inform user that he is controlling the platform
```

Figure A9: PIC Basic code of the user interface (Continued)

```
    while(pushButton == 1) ' Wait until button has been pushed again to exit
subroutine
        high bitA
        low bitB
    wend

return


accel_LED:

    while(pushButton == 0) ' Wait until button is released to proceed
    wend


    Lcdout $fe, 1    ' Clear LCD screen
    Lcdout "LED Hand Control"  ' Inform user that he is controlling the High
Power LED with his wrist




    while(pushButton == 1) ' Wait until button has been pushed again to exit
subroutine
        low bitA
        high bitB
    wend
return


pot_LED:

    while(pushButton == 0) ' Wait until button is released to proceed
    wend

    Lcdout $fe, 1    ' Clear LCD screen
    Lcdout "LED Pot Control"  ' Inform user that he is controlling the High
Power LED with the pot




    while(pushButton == 1) ' Wait until button has been pushed again to exit
subroutine
        high bitA
        high bitB
    wend

return

End
```

Figure A9: PIC Basic code of the user interface (Continued)

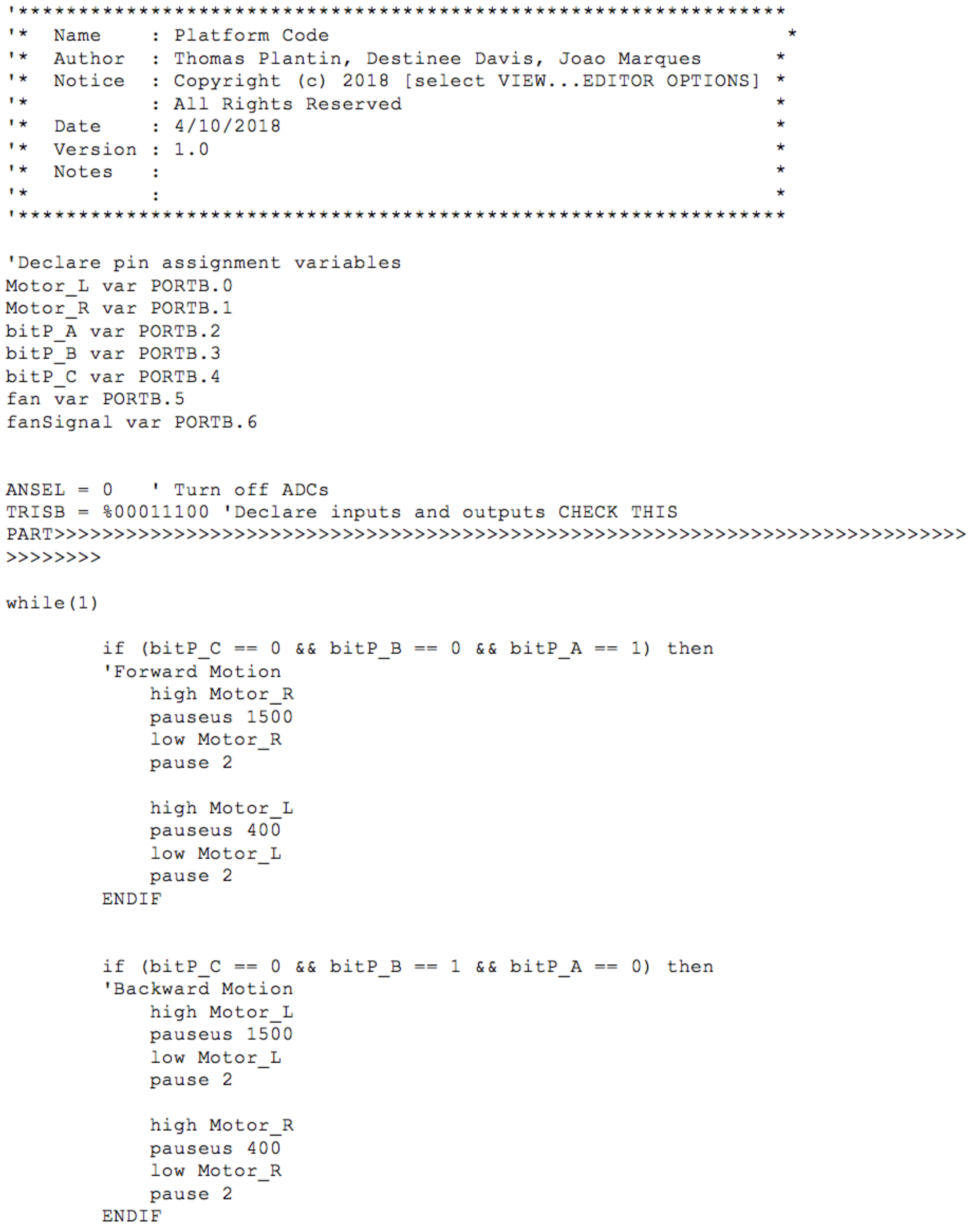


```
'*****************************************************************
'*  Name    : Global Control Scheme                                    *
'*  Author  : Thomas, Destinee, Joao              *
'*  Notice  : Copyright (c) 2018 [select VIEW...EDITOR OPTIONS] *
'*          : All Rights Reserved                               *
'*  Date    : 4/2/2018                                        *
'*  Version : 1.0                                               *
'*  Notes   :                                                   *
'*          :                                                   *
'*****************************************************************

DEFINE  ADC_BITS        10      ' Set number of bits in result
DEFINE  ADC_CLOCK       3       ' Set clock source (3=rc)
DEFINE  ADC_SAMPLEUS    15      ' Set sampling time in uS

' Set inputs (1) and outputs (0) of PORTA
TRISA = %11111111

' Set up ADCON1
ADCON1 = %10000000 ' Right-justify results (lowest 10 bits)

' Declare pin assignment variables
bitA var PORTB.0
bitB var PORTB.1
HpLED var PORTB.4
potentiometer con 2

' Declare logic variables
potValue var word
strobeFlash var word
onTime con 200



low HpLED



while(bitA == 1 && bitB == 0) ' Platform control


wend




while(bitA == 0 && bitB == 1) ' Hand accelerometer LED control

wend
```

Figure A10: PIC Basic code of the potentiometer controlled LED flashing

```
while(bitA == 1 && bitB == 1)  'Potentiometer LED control

    adcin potentiometer, potValue ' Read the potentiometer value
    strobeFlash = ((2046-potValue)+100*(potValue))/(99)

        high HpLED
        pauseus onTime
        low HpLED
        pause strobeFlash
wend
```

Figure A10: PIC Basic code of the potentiometer controlled LED flashing (Continued)

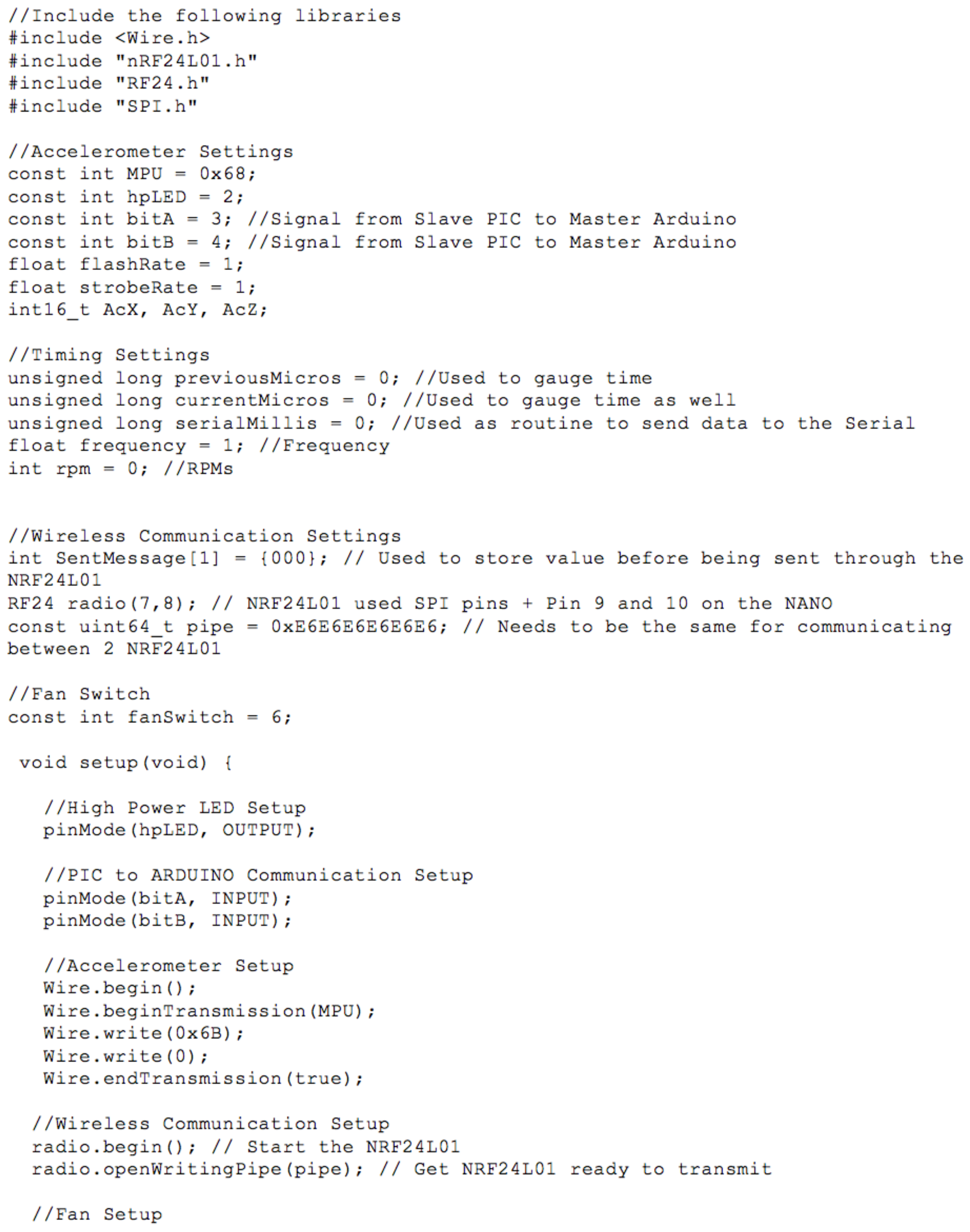


```
'*****************************************************************
'*  Name    : Platform Code                                       *
'*  Author  : Thomas Plantin, Destinee Davis, Joao Marques      *
'*  Notice  : Copyright (c) 2018 [select VIEW...EDITOR OPTIONS] *
'*          : All Rights Reserved                               *
'*  Date    : 4/10/2018                                         *
'*  Version : 1.0                                               *
'*  Notes   :                                                   *
'*          :                                                   *
'*****************************************************************

'Declare pin assignment variables
Motor_L var PORTB.0
Motor_R var PORTB.1
bitP_A var PORTB.2
bitP_B var PORTB.3
bitP_C var PORTB.4
fan var PORTB.5
fanSignal var PORTB.6


ANSEL = 0   ' Turn off ADCs
TRISB = %00011100 'Declare inputs and outputs CHECK THIS
PART>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>

while(1)

        if (bitP_C == 0 && bitP_B == 0 && bitP_A == 1) then
        'Forward Motion
            high Motor_R
            pauseus 1500
            low Motor_R
            pause 2

            high Motor_L
            pauseus 400
            low Motor_L
            pause 2
        ENDIF


        if (bitP_C == 0 && bitP_B == 1 && bitP_A == 0) then
        'Backward Motion
            high Motor_L
            pauseus 1500
            low Motor_L
            pause 2

            high Motor_R
            pauseus 400
            low Motor_R
            pause 2
        ENDIF
```

Figure A11: PIC Basic code of the platform control

```
        if (bitP_C == 0 && bitP_B == 1 && bitP_A == 1) then
        'Leftward Motion
            high Motor_L
            pauseus 400
            low Motor_L
            pause 2

            high Motor_R
            pauseus 400
            low Motor_R
            pause 2
        ENDIF

        if (bitP_C == 1 && bitP_B == 0 && bitP_A == 0) then
        'Rightward Motion
            high Motor_L
            pauseus 1500
            low Motor_L
            pause 2

            high Motor_R
            pauseus 1500
            low Motor_R
            pause 2
        ENDIF

        if (bitP_C == 0 && bitP_B == 0 && bitP_A == 0) then
        'Idle mode
            low Motor_R
            low Motor_L
        ENDIF

        if (fanSignal == 1) then
        'Fan on
            high fan
        ENDIF

        if (fanSignal == 0) then
        'Fan off
            low fan
        endif

wend
```

Figure A11: PIC Basic code of the platform control (Continued)

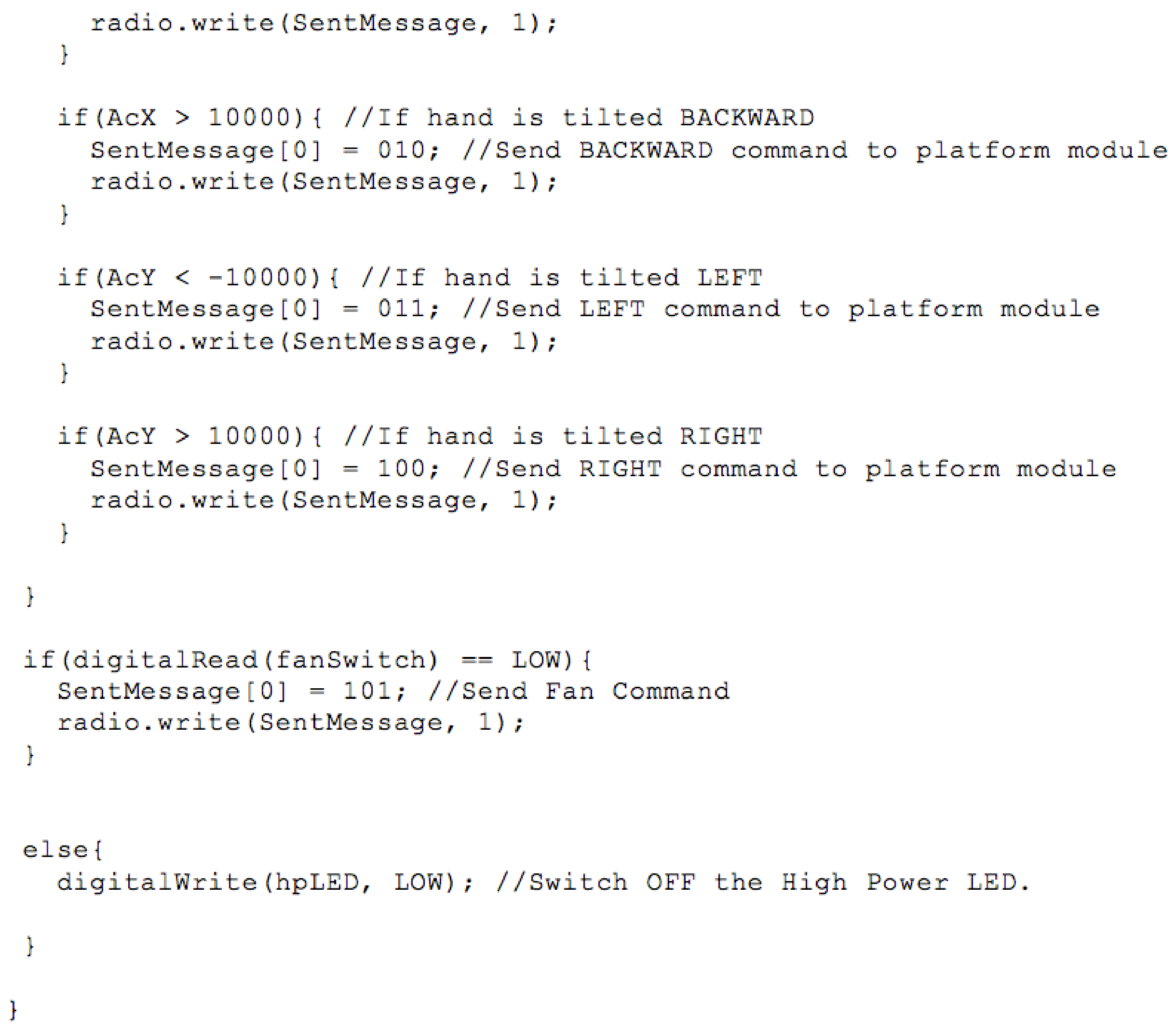


```
//Include the following libraries
#include <Wire.h>
#include "nRF24L01.h"
#include "RF24.h"
#include "SPI.h"

//Accelerometer Settings
const int MPU = 0x68;
const int hpLED = 2;
const int bitA = 3; //Signal from Slave PIC to Master Arduino
const int bitB = 4; //Signal from Slave PIC to Master Arduino
float flashRate = 1;
float strobeRate = 1;
int16_t AcX, AcY, AcZ;

//Timing Settings
unsigned long previousMicros = 0; //Used to gauge time
unsigned long currentMicros = 0; //Used to gauge time as well
unsigned long serialMillis = 0; //Used as routine to send data to the Serial
float frequency = 1; //Frequency
int rpm = 0; //RPMs


//Wireless Communication Settings
int SentMessage[1] = {000}; // Used to store value before being sent through the
NRF24L01
RF24 radio(7,8); // NRF24L01 used SPI pins + Pin 9 and 10 on the NANO
const uint64_t pipe = 0xE6E6E6E6E6E6; // Needs to be the same for communicating
between 2 NRF24L01

//Fan Switch
const int fanSwitch = 6;

 void setup(void) {

   //High Power LED Setup
   pinMode(hpLED, OUTPUT);

   //PIC to ARDUINO Communication Setup
   pinMode(bitA, INPUT);
   pinMode(bitB, INPUT);

   //Accelerometer Setup
   Wire.begin();
   Wire.beginTransmission(MPU);
   Wire.write(0x6B);
   Wire.write(0);
   Wire.endTransmission(true);

  //Wireless Communication Setup
  radio.begin(); // Start the NRF24L01
  radio.openWritingPipe(pipe); // Get NRF24L01 ready to transmit

  //Fan Setup
```

Figure A12: Arduino code of the transmitter / hand module

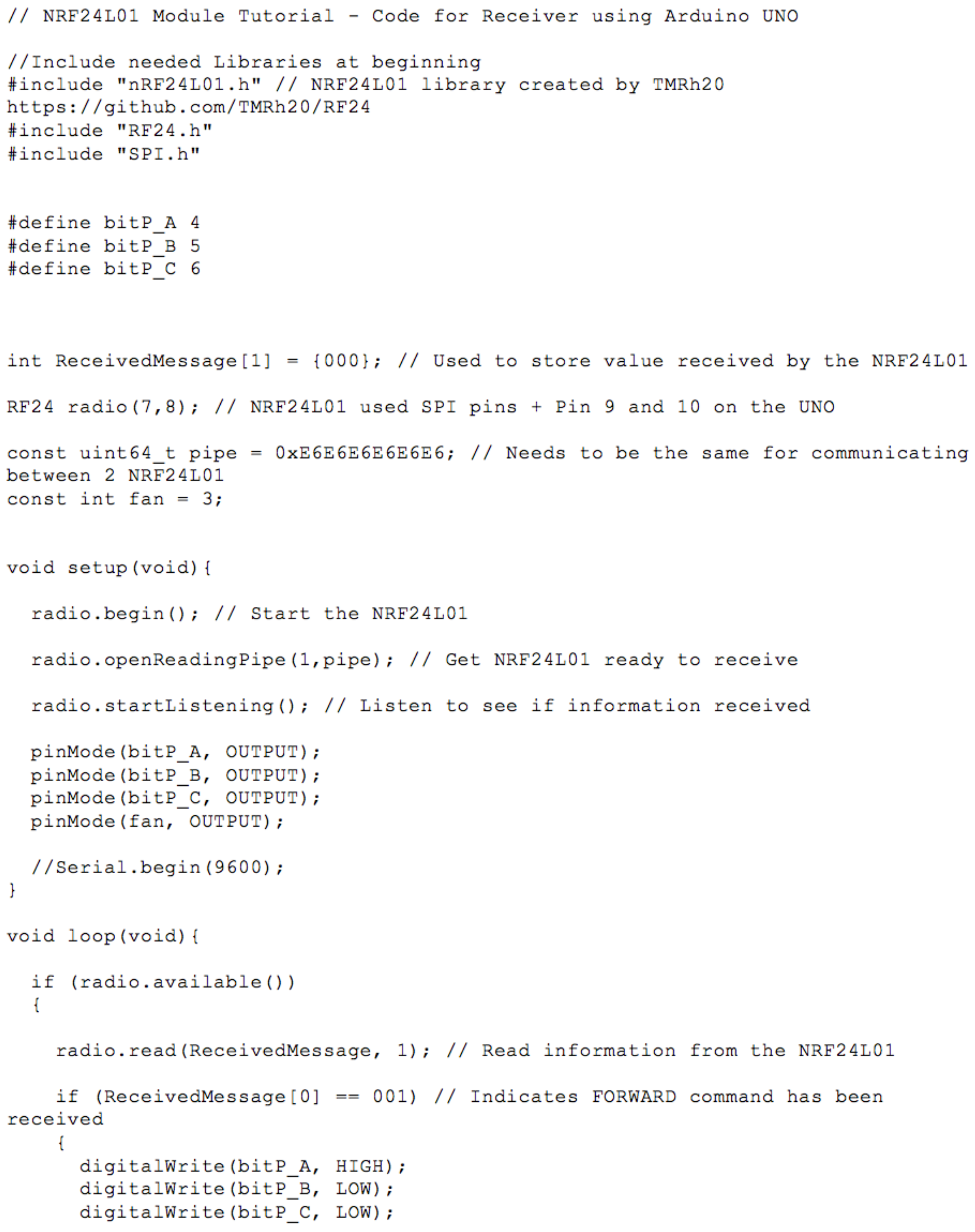
```
  pinMode(fanSwitch, INPUT_PULLUP);
  digitalWrite(fanSwitch, HIGH);

  Serial.begin(9600);
 }



 void loop(void) {
  Wire.beginTransmission(MPU);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU, 12, true);
  AcX = Wire.read()<<8|Wire.read();
  AcY = Wire.read()<<8|Wire.read();
  AcZ = Wire.read()<<8|Wire.read();
  //Serial.print("X_Acceleration = "); Serial.print(AcX);
  //Serial.print(" || Y_Acceleration = "); Serial.print(AcY);
  //Serial.print(" || Z_Acceleration = "); Serial.println(AcZ);


  if(digitalRead(bitA) == LOW && digitalRead(bitB) == HIGH){ //Activates LED
control.

    SentMessage[0] = 000; //Set platform to iddle mode.
    radio.write(SentMessage, 1);

   if(AcX > 7000){ //If hand is lifted
    flashRate = map(AcY, -10000, 13000, 0, 1023); //Measure and map
Accelerometer y-direction value
    strobeRate = flashRate*54 + 400; //Duration of the stroboscopic pusle >
empirical method
    currentMicros = micros(); //Registration of the current time of the timer.

    if(currentMicros > previousMicros + strobeRate){
      digitalWrite(hpLED, HIGH);
      delayMicroseconds(200);
      digitalWrite(hpLED, LOW);
      //Serial.println("ON");
    }
   }


   else{
    digitalWrite(hpLED, LOW);
   }

  }

  if(digitalRead(bitA) == HIGH && digitalRead(bitB) == LOW){ //Activates Plaform
control.

    if(AcX < -10000){ //If hand is tilted FORWARD
      SentMessage[0] = 001; //Send FORWARD command to platform module
```

Figure A12: Arduino code of the transmitter / hand module (Continued)



```
      radio.write(SentMessage, 1);
    }

    if(AcX > 10000){ //If hand is tilted BACKWARD
      SentMessage[0] = 010; //Send BACKWARD command to platform module
      radio.write(SentMessage, 1);
    }

    if(AcY < -10000){ //If hand is tilted LEFT
      SentMessage[0] = 011; //Send LEFT command to platform module
      radio.write(SentMessage, 1);
    }

    if(AcY > 10000){ //If hand is tilted RIGHT
      SentMessage[0] = 100; //Send RIGHT command to platform module
      radio.write(SentMessage, 1);
    }

  }

  if(digitalRead(fanSwitch) == LOW){
    SentMessage[0] = 101; //Send Fan Command
    radio.write(SentMessage, 1);
  }


  else{
    digitalWrite(hpLED, LOW); //Switch OFF the High Power LED.

  }

}
```

Figure A12: Arduino code of the transmitter / hand module (Continued)



```
// NRF24L01 Module Tutorial - Code for Receiver using Arduino UNO

//Include needed Libraries at beginning
#include "nRF24L01.h" // NRF24L01 library created by TMRh20
https://github.com/TMRh20/RF24
#include "RF24.h"
#include "SPI.h"


#define bitP_A 4
#define bitP_B 5
#define bitP_C 6



int ReceivedMessage[1] = {000}; // Used to store value received by the NRF24L01

RF24 radio(7,8); // NRF24L01 used SPI pins + Pin 9 and 10 on the UNO

const uint64_t pipe = 0xE6E6E6E6E6E6; // Needs to be the same for communicating
between 2 NRF24L01
const int fan = 3;


void setup(void){

  radio.begin(); // Start the NRF24L01

  radio.openReadingPipe(1,pipe); // Get NRF24L01 ready to receive

  radio.startListening(); // Listen to see if information received

  pinMode(bitP_A, OUTPUT);
  pinMode(bitP_B, OUTPUT);
  pinMode(bitP_C, OUTPUT);
  pinMode(fan, OUTPUT);

  //Serial.begin(9600);
}

void loop(void){

  if (radio.available())
  {

    radio.read(ReceivedMessage, 1); // Read information from the NRF24L01

    if (ReceivedMessage[0] == 001) // Indicates FORWARD command has been
received
    {
      digitalWrite(bitP_A, HIGH);
      digitalWrite(bitP_B, LOW);
      digitalWrite(bitP_C, LOW);
```

Figure A13: Arduino code of the receiver / platform module

```
      //Serial.println("FORWARD");
    }

    if (ReceivedMessage[0] == 010) // Indicates BACKWARD command has been
received
    {
      digitalWrite(bitP_A, LOW);
      digitalWrite(bitP_B, HIGH);
      digitalWrite(bitP_C, LOW);

      //Serial.println("BACKWARD");
    }

    if (ReceivedMessage[0] == 011) // Indicates LEFT command has been received
    {
      digitalWrite(bitP_A, HIGH);
      digitalWrite(bitP_B, HIGH);
      digitalWrite(bitP_C, LOW);

      //Serial.println("LEFT");
    }

    if (ReceivedMessage[0] == 100) // Indicates RIGHT command has been received
    {
      digitalWrite(bitP_A, LOW);
      digitalWrite(bitP_B, LOW);
      digitalWrite(bitP_C, HIGH);

      //Serial.println("RIGHT");
    }

    if(ReceivedMessage[0] == 101) //FanON command received.
    {
      digitalWrite(fan, HIGH);
    }

    if(ReceivedMessage[0] != 101) //FanOFF command received.
    {
      digitalWrite(fan, LOW);
    }
 }

 else // Indicates that NO command has been received. So set platform IDLE.
    {
      digitalWrite(bitP_A, LOW);
      digitalWrite(bitP_B, LOW);
      digitalWrite(bitP_C, LOW);

      //Serial.println("IDLE");

    }
  delay(10);
}
```

Figure A13: Arduino code of the receiver / platform module (Continued)

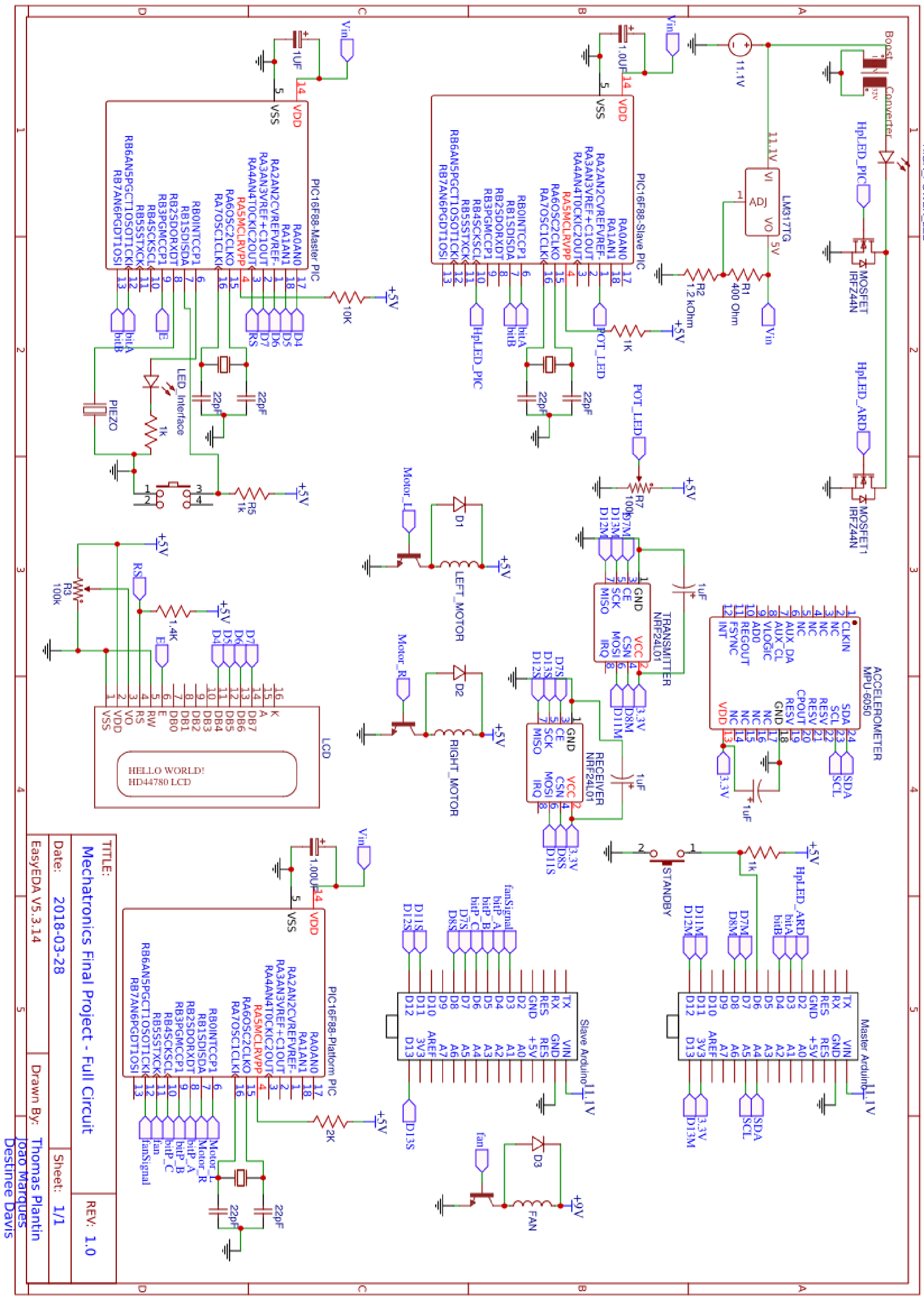


Figure A14: Circuit schematic of the final prototype