Hardware Setup

The LCD shield used is the **LCD Keypad Shield v1.1 (black)**.

The following pins were used for the motor:

MOTOR PIN	ARDUINO PIN
IN1	D13
IN2	D12
IN3	D11
IN4	D3

The IR sensor is read through Analog Pin 1 on the Arduino.

The IR sensor was purchased from the website given in the beginning of the semester and the model number is GP2Y0A02YK0F.

Code

```
#include <LiquidCrystal.h>

#include <avr/io.h>

#include <avr/interrupt.h>



LiquidCrystal lcd(8, 9, 4, 5, 6, 7); // Set up LCD


volatile unsigned long int milliseconds = 0;

volatile long unsigned int timer;

boolean blinking = true;



/*** IR sensor variables ***/

volatile float irHistory[5];

volatile int irValue =0;


/*** MOTOR VARIABLES ***/

int Steps;

int Direction = 0;

int StepTime = 2;
int previousStepTime = 0;

int motorSpeed = 2;

boolean startMotor = false;

int wheelSize = 20;

int stepSet = 100;

int stepCount = 0;

unsigned int stepRemaining = 0;



/*** MODE ***/

enum Mode {

  startupMODE,

  debugMODE,

  irMODE,

  cmMODE,

  pmMODE,

  setMODE,

  driveMODE

};

Mode currentMode;


/*** State for flickering state for menu selection ***/

enum debugState {

  IR,

  CM,

  PM,

  SET,

  E

};

debugState currentDebugState = IR;
```

```
enum cmState {
  starting,
  exiting
};
cmState currentCMState = starting;

enum driveModeState {
  idle,
  CW,
  CCW
};
driveModeState currentDriveState = idle;

/** Button inputs **/
enum Buttons {
  btnNONE,
  btnRIGHT,
  btnUP,
  btnDOWN,
  btnLEFT,
  btnSELECT
};


Buttons whatbuttons; // Current button pressed
Buttons buttonHistory[5] = { btnNONE, btnNONE, btnNONE, btnNONE, btnNONE };
Buttons debugSequence[5] = { btnLEFT, btnLEFT, btnUP, btnRIGHT, btnSELECT }; // to get into debug mode

volatile int motorDelay = 0;
```

```
ISR(TIMER2_OVF_vect) { //Chapter 16
  //Register size = 64
  // CLK = 62500 Hz
  //Timer pertick = 1/CLK = 0.016ms
  // from 0 to 64 = 64 * 0.016ms = 1ms

  milliseconds += 1; //increment every ms



  if(startMotor) { // one step every ms for drive mode
    if(stepRemaining != 0) { // used for PM and drive MODE
      stepperMotor(1);
      stepRemaining--;
    }
    if(currentMode == cmMODE) { // For CM Mode only
      motorDelay++; // counter works as a delay for the different speeds
      if(motorDelay >= motorSpeed) { // if the counter is higher than motorSpeed, one step is done
        stepperMotor(1);
        motorDelay = 0;
      }
    }

  }



}



void setup() {
  Serial.begin(9600);
```

```
 timer_Init();
 lcd_Init();
 ADC_Init();
 stepperMotor_Init();
 currentMode = startupMODE;


}


void loop() {
 switch(currentMode) {
  case startupMODE:
   printClock(mymillis()/60000, (mymillis() / 1000) % 60); //Convert the values to the print the clock
   if(checkDebugSequence()) { // check debug sequence, if it is right, go to debug sequence
    currentMode = debugMODE;


   }
   else if ((whatbuttons == btnSELECT) && !(checkDebugSequence())) {
    currentMode = driveMODE; //goes drive mode if it doesnt equal sequence
   }
   break;
  case debugMODE:
   debugModeOperation();
   break;
  case irMODE:
   irModeOperation();
   break;
  case cmMODE:
   cmModeOperation();
   break;
  case pmMODE:
   pmModeOperation();
```

```
      break;
    case setMODE:
      setModeOperation();
      break;
    case driveMODE:
      driveModeOperation();
      break;
    default:
      break;
  }

  buttonSlidingWindow(); // for the button history

//  if(whatbuttons != btnNONE) {
//    Serial.println(whatbuttons);
//  }

}



void timer_Init() {
  //Timer2 register A = normal operation
  TCCR2A &=~(1<<WGM20);
  TCCR2A &=~(1<<WGM21);
  TCCR2A &=~(1<<COM2B0);
  TCCR2A &=~(1<<COM2B1);
  TCCR2A &=~(1<<COM2A0);
  TCCR2A &=~(1<<COM2A1);

  // Prescaler for 64
  TCCR2B &= ~(1<<CS20);
  TCCR2B &= ~(1<<CS21);
  TCCR2B |= (1<<CS22);

  TIMSK2 |= (1<<TOIE2); //Enable Overflow interrupt
  sei(); //enable global interrupt
  TIFR2 |= (1<<TOV2);
}

volatile unsigned long int mymillis() {
  return milliseconds;
}

void mydelay(volatile long unsigned int delayTime) {

  volatile long unsigned int count = mymillis();
  while(mymillis() <= (delayTime + count)) {

  }
}

void printClock(int minutes, int seconds) { // print clock for startup mode
  lcd.setCursor(0,1);
  lcd.print("ID:12878930");
  lcd.setCursor(0,0);
  if(minutes <10) {
    lcd.print('0');
  }
  lcd.print(minutes);
  lcd.print(':');
  if(seconds < 10) {
    lcd.print('0');
  }
  lcd.print(seconds);
}

void lcd_Init() {
  lcd.begin(16,2);
  lcd.clear();
}

Buttons readLCDButtons() {
  static int inputButton;
  inputButton = myAnalogRead(0);
  mydelay(145); //DEBOUNCE
  // read analog 0 with registers
  if(inputButton > 1000) {
    return btnNONE;
  }
  if(inputButton < 50) {
    return btnRIGHT;
  }
  if(inputButton < 250) {
    return btnUP;
  }
  if(inputButton < 450) {
    return btnDOWN;
  }
  if(inputButton < 650) {
    return btnLEFT;
  }
  if(inputButton < 850) {
    return btnSELECT;
  }


  return btnNONE; // when all others fail, return this
}


void buttonSlidingWindow() {
  whatbuttons = readLCDButtons();
  if(whatbuttons != btnNONE) {
    for(int i = 0; i < 4; i++) {
      buttonHistory[i] = buttonHistory[i+1];
    }
    buttonHistory[4] = whatbuttons;
  }
}


boolean checkDebugSequence() {
  for(int i = 0; i <= 4; i++) {
    if(buttonHistory[i] != debugSequence[i]) {
      return false;
    }
  }
  return true;
}



void ADC_Init() {
  ADCSRA |= (1<<ADEN); // Enable ADC
  ADMUX |= (1<<REFS0); // Internal Vcc 5v
}
```

```
int myAnalogRead(int Pin)
{
 if(Pin == 1) {
   ADMUX |= Pin; //Multiplexer  for which pin to read from
 }
 else if(Pin == 0) {
   ADMUX = 0;
   ADMUX |= (1<<REFS0); // Internal Vcc 5v
 }
 ADCSRA |= (1<<ADSC); // start conversion
 // wait for conversion to complete
 while (!(ADCSRA &(1<<ADIF))); // becomes while(0) when the conversion is complete
 ADCSRA |= (1<<ADIF);
 return ADC;
}



void stepperMotor_Init() {
 DDRB |= (1<<DDB5); // Change data direction of pin 13 to output
 DDRB |= (1<<DDB4); // Change data direction of pin 12 to output
 DDRB |= (1<<DDB3); // Change data direction of pin 11 to output
 DDRD |= (1<<DDD3); // Change data direction of pin 3 to output


}


void stepperMotor(int xw) {
 for(int x =0; x< xw; x++)
 {
   switch(Steps)
   {
     case 0: //1000
     PORTB |= (1<<PORTB5);
     PORTB &= ~(1<<PORTB4);
     PORTB &= ~(1<<PORTB3);
     PORTD &= ~(1<<PORTD3);
     break;
    case 1: //1100
     PORTB |= (1<<PORTB5);
     PORTB |= (1<<PORTB4);
     PORTB &= ~(1<<PORTB3);
     PORTD &= ~(1<<PORTD3);
     break;
    case 2: //0100
     PORTB &= ~(1<<PORTB5);
     PORTB |= (1<<PORTB4);
     PORTB &= ~(1<<PORTB3);
     PORTD &= ~(1<<PORTD3);
     break;
    case 3: //0110
     PORTB &= ~(1<<PORTB5);
     PORTB |= (1<<PORTB4);
     PORTB |= (1<<PORTB3);
     PORTD &= ~(1<<PORTD3);
     break;
    case 4: //0010
     PORTB &= ~(1<<PORTB5);
     PORTB &= ~(1<<PORTB4);
     PORTB |= (1<<PORTB3);
     PORTD &= ~(1<<PORTD3);
     break;
    case 5: //0011
     PORTB &= ~(1<<PORTB5);
```

```
        PORTB &= ~(1<<PORTB4);
        PORTB |= (1<<PORTB3);
        PORTD |= (1<<PORTD3);
        break;
      case 6: //0001
        PORTB &= ~(1<<PORTB5);
        PORTB &= ~(1<<PORTB4);
        PORTB &= ~(1<<PORTB3);
        PORTD |= (1<<PORTD3);
        break;
      case 7: //1001
        PORTB |= (1<<PORTB5);
        PORTB &= ~(1<<PORTB4);
        PORTB &= ~(1<<PORTB3);
        PORTD |= (1<<PORTD3);
        break;
      default: //0000
        PORTB &= ~(1<<PORTB5);
        PORTB &= ~(1<<PORTB4);
        PORTB &= ~(1<<PORTB3);
        PORTD &= ~(1<<PORTD3);
        break;
    }
    setDirection();
  }
}

void setDirection() {
  if(Direction ==1) { //anticlockwise
    Steps++;
  }
  else { //clockwise
    Steps--;
  }
  if(Steps > 7) {
    Steps = 0;
  }
  if(Steps < 0) {
    Steps = 7;
  }
}

void debugModeOperation() {
  lcd.setCursor(0,0);
  lcd.print("DEBUG Mode");

  if(blinking) {
    lcd.setCursor(0,1);
    lcd.print("IR CM PM SET E");
    timer = mymillis();
    blinking = false;
  }

  if(mymillis() >= (timer + 1000)) { // Blinks every second for the cursor
    blinking = true;
    lcd.setCursor(0,1);
    switch(currentDebugState) {
      case IR:
        lcd.print("   CM PM SET E");
        break;
      case CM:
        lcd.print("IR    PM SET E");
    break;
   case PM:
    lcd.print("IR CM    SET E");
    break;
   case SET:
    lcd.print("IR CM PM     E");
    break;
   case E:
    lcd.print("IR CM PM SET  ");
    break;
   default:
    break;
  }
 }

 switch(whatbuttons) { //User inputs
  case btnLEFT: // cursor going to the left
   currentDebugState = currentDebugState - 1;
   if(currentDebugState < 0) {
    currentDebugState = 0;
   }
   break;
  case btnRIGHT:  // cursor going to the right
   currentDebugState = currentDebugState +1;
   if(currentDebugState > 4) {
    currentDebugState = 4;
   }
   break;
  case btnSELECT: // selecting which mode
   debugModeSelection();
   blinking = true;
```

```
      currentDebugState = IR;
      lcd.clear();
      break;
  }

}

void debugModeSelection() {
  switch(currentDebugState) {
    case IR:
      currentMode = irMODE;
      break;
    case CM:
      currentMode = cmMODE;
      break;
    case PM:
      currentMode = pmMODE;
      break;
    case SET:
      currentMode = setMODE;
      break;
    case E:
      currentMode = startupMODE;
      break;
    default:
      break;
  }
}
```

```
void driveModeOperation() {
  static int averaging =0; // averaging of the IR value
  static volatile float sum =0;
  static volatile float sensorValue =0;
  static long unsigned int previousTime = 0;
  static float revolutions = 0;
  static boolean startupCondition = true;



  if(startupCondition) {
    lcd.clear();
    startupCondition = false;
  }

  lcd.setCursor(0,0);
  lcd.print("Drive Mode");

  switch(currentDriveState) {
    case idle: //idle means that motor isnt running
      if(averaging < 5) {
        if(mymillis() - previousTime >= 200) {
          averaging++;
          sensorValue = myAnalogRead(1);
          mydelay(20);
          sensorValue = 55.55/(sensorValue*5/1023);
          sum += sensorValue; // found from excel line of best fit + datasheet
          previousTime = mymillis();
        }
      }
      else if(averaging >= 5){ //Averaging after reading  values
          irValue = sum/averaging;
```

```
     if(irValue > 150) {
      irValue = 150;
     }
     lcd.setCursor(0,1);
     lcd.print("                ");
     lcd.setCursor(0,1);
     lcd.print(irValue);
     lcd.print(" ");
     revolutions = (float) irValue/ (float) wheelSize; // revolutions = distance / circumference
     lcd.print(revolutions, 1);
     lcd.print(" ");
     stepRemaining = revolutions * 4096; // one revolution is 4096 steps
     lcd.print(stepRemaining);
     averaging =0;
     sum = 0;
   }
   break;
  case CW: //clockwise rotation
    lcd.setCursor(0,1);
    lcd.print("                ");
    lcd.setCursor(0,1);
    lcd.print(irValue);
    lcd.print(" ");
    lcd.print(revolutions, 1);
    lcd.print(" ");
    lcd.print(stepRemaining);
    if(stepRemaining == 0) {
     startMotor = false;
     motorClear(); // clear all outputing leds of motor
     currentDriveState = idle;
    }
```

```
      break;
    case CCW: //anticlockwise rotation
        lcd.setCursor(0,1);
        lcd.print("                ");
        lcd.setCursor(0,1);
        lcd.print(irValue);
        lcd.print(" ");
        lcd.print(revolutions, 1);
        lcd.print(" ");
        lcd.print(stepRemaining);
        if(stepRemaining == 0) {
          startMotor = false;
          currentDriveState = idle;
        }
      break;
  }

  switch(whatbuttons) {
    case btnSELECT:
      // clear all non initial state
      currentMode = startupMODE;
      Direction = 0;
      currentDriveState = idle;
      startupCondition = true;
      startMotor = false;
      stepRemaining = 0;
      motorClear();
      lcd.clear();

      break;
    case btnUP:
```

```
    if(!startMotor) { // if motor is running, dont do anything
      startMotor = true;
      Direction =0;
      currentDriveState = CW;
      stepRemaining = revolutions*4096;
    }
      break;
    case btnDOWN:
      if(!startMotor){
        startMotor = true;
        Direction = 1;
        currentDriveState = CCW;
        stepRemaining = revolutions*4096;
      }
      break;
    default:
      break;
  }
}



void irModeOperation() {
  static int averaging = 0;
  static volatile float sum = 0;
  static volatile float sensorValue = 0;
  static long unsigned int previousTime = 0;

  lcd.setCursor(0,0);
  lcd.print("IR Mode");

  if(averaging < 5) { // averaging every 5 values found
```

```
    if(mymillis() - previousTime >= 200) {
      averaging++;
      sensorValue = myAnalogRead(1);
      mydelay(20);
      sensorValue = 55.55/(sensorValue*5/1023);
      sum += sensorValue; // found from excel line of best fit + datasheet
      previousTime = mymillis();
    }
  }
  else if(averaging >= 5){
      irValue = sum/averaging;
      if(irValue > 150) {
        irValue = 150;
      }
      lcd.setCursor(0,1);
      lcd.print("      ");
      lcd.setCursor(0,1);
      lcd.print(irValue);
      lcd.print(" cm");
      averaging =0;
      sum = 0;
  }



  if(whatbuttons == btnSELECT) {
    lcd.clear();
    averaging =0;
    sum = 0;
    currentMode = debugMODE;
  }
```

```
}



void motorClear() { // clearing all the ports used by the motor - if not conflicts with IR

  PORTB &= ~(1<<PORTB5);
  PORTB &= ~(1<<PORTB4);
  PORTB &= ~(1<<PORTB3);
  PORTD &= ~(1<<PORTD3);
}

void setModeOperation() {
  lcd.setCursor(0,0);
  lcd.print("SETTINGS Mode");
  lcd.setCursor(0,1);
  lcd.print("Wheel: ");
  lcd.setCursor(7,1);
  lcd.print(wheelSize);
  lcd.setCursor(10, 1);
  lcd.print("cm");

  switch(whatbuttons) {
    case btnUP:
      wheelSize += 10;
      if(wheelSize > 90) { // Upper limit of the wheel is 90cm circumference
        wheelSize = 90;
      }
      break;
    case btnDOWN:
      wheelSize -= 10;
      if(wheelSize < 10) {
        wheelSize = 10;
      }
      break;
    case btnSELECT:
      lcd.clear();
      currentMode = debugMODE;
      break;
  }
}

void pmModeOperation() {
  static boolean pmStart = false;
  lcd.setCursor(0,0);
  lcd.print("PM Mode");
  lcd.setCursor(0,1);
  lcd.print("                ");
  lcd.setCursor(0,1);
  lcd.print(stepSet);
  lcd.print(" ");
  lcd.print(stepSet);

  if(!pmStart) { // when the motor isnt running yet
    switch(whatbuttons) {
      case btnUP:
        stepSet += 100;
        if(stepSet > 30000) {
          stepSet = 30000;
        }
        break;
      case btnDOWN:
```

```
    stepSet -= 100;
    if(stepSet < 100) {
     stepSet = 100;
    }
    break;
   case btnLEFT:
    stepSet = 100;
    break;
   case btnRIGHT:
    pmStart = true;
    startMotor = true;
    stepRemaining = stepSet;
    break;
   case btnSELECT:
    lcd.clear();
    currentMode = debugMODE;
  }
 }
 else {

  lcd.setCursor(0,1);
  lcd.print("            ");
  lcd.setCursor(0,1);
  lcd.print(stepSet);
  lcd.print(" ");
  lcd.print(stepRemaining);

  if(stepRemaining == 0) {
   pmStart = false; //if no more step, goes back to idle state of pm mode
   motorClear(); // clear the output LEDs motor
  }
```

```
    if(whatbuttons == btnSELECT) {
      lcd.clear();
      pmStart = false;
      stepRemaining = 0;
      motorClear();
      currentMode = debugMODE;
    }

  }


}


void cmModeOperation() {
  lcd.setCursor(0,0);
  lcd.print("CM Mode");
  if(blinking) {
    lcd.setCursor(0,1);
    lcd.print("Start Exit");
    timer = mymillis();
    blinking = false;
  }


  // blinking used again
  if(mymillis() >= (timer + 1000) && startMotor == false) {
    blinking = true;
    lcd.setCursor(0,1);
    switch(currentCMState) {
      case starting:
        lcd.print("       Exit");
        break;
      case exiting:
        lcd.print("Start     ");
        break;
    }
  }

  if(!startMotor) { // when idle
    switch(whatbuttons) {
      case btnLEFT:
        currentCMState = starting;
        break;
      case btnRIGHT:
        currentCMState = exiting;
        break;
      case btnSELECT:
        if(currentCMState == exiting) {
          blinking = true;
          currentCMState = starting;
          lcd.clear();
          currentMode = debugMODE;
        }
        else {
          startMotor = true;
          blinking = false;
          lcd.clear();
          lcd.setCursor(0,0);
          lcd.print("CM Mode");
        }
        break;
    }
  }
  else { // when motor is started
    lcd.setCursor(0,1);
    if(Direction == 0) {
      lcd.print("CW");
      lcd.print(" Speed: ");
      switch(motorSpeed) { // motorSpeed is how much delay
        case 1:
          lcd.print("fast");
          break;
        case 2:
          lcd.print("medium");
          break;
        case 3:
          lcd.print("slow");
          break;
      }
    }
    else {
      lcd.print("CCW");
      lcd.print(" Speed: ");
      switch(motorSpeed) {
        case 1:
          lcd.print("fast");
          break;
        case 2:
          lcd.print("medium");
```

```
     break;
    case 3:
     lcd.print("slow");
     break;
   }
  }
  switch(whatbuttons) {
   case btnLEFT: //anticlockwise
    Direction = 1;
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("CM Mode");
    break;
   case btnRIGHT: // clockwise
    Direction = 0;
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("CM Mode");
    break;
   case btnUP:
    motorSpeed--; //decrease the delay will increase the speed
    if(motorSpeed < 1) {
     motorSpeed = 1;
    }
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("CM Mode");
    break;
   case btnDOWN:
    motorSpeed++; // increase the delay to decrease the speed
    if(motorSpeed > 3) {
          motorSpeed = 3;
        }
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("CM Mode");
        break;
      case btnSELECT:
        motorSpeed = 2; // back to initial stage
        Direction = 0;
        startMotor = false;
        motorClear();
        blinking = true;
        lcd.clear();
        break;
    }

  }

}
```