University of Washington
EE 299 Lab 1
Introducing the Lab Environment

Mitchell Szeto
Bert Zhao
Feifan Qiao

TABLE OF CONTENTS

ABSTRACT

In this two part lab, we explored the C language, Arduino IDE, Arduino UNO microprocessor, and several microprocessor peripherals. This lab guided us through many different programs that gradually became more complex--to the point where we were able to set up an I2C connection between two Arduinos. This lab highlights the foundation for embedded programming by using the Arduino UNO microcontroller. We will build on what we have learned in this lab as we complete projects in future labs.

INTRODUCTION

This lab is split into two major parts. There are three goals of the first part of the lab. The first is to introduce the C language and the Arduino IDE. We explore some of the basic functions of the language within this lab. The second goal is to introduce the Arduino UNO microprocessor. We learn to use the Arduino IDE to execute sample programs on the Arduino UNO microprocessor through some simple C functions. The third objective is to work with some of the sample programs within the Arduino IDE. We change the sample blink program to work with other peripheral devices such as the push button, and LEDs. We also work with the serial monitor as a secondary display.

The second part of this lab focuses on peripheral displays and simple networks. We work with the LCD and push buttons to manipulate our output to the LCD. We explore a simple network by connecting two of the Arduino UNO microprocessors. There is a master and a slave and we explore how to transmit and display information between them.

DISCUSSION OF LAB

Venturing Out - First Steps

Design Specification

In the first steps, we were tasked to write two programs that could manipulate the Onboard LED. In Program 1, the Onboard LED had to be on for 2 seconds and off for 1 second. In Program 2, the Onboard LED had two 1 seconds blinks and is shortly followed by two 2 second blinks.

Hardware Implementation

Since there were no peripherals involved the only hardware and connections that we used were the Arduino Uno and the USB connection to a computer.

Software Implementation

The software design for Program 1 and 2 are procedural. We simply wrote the instructions to for the arduino to turn on the Onboard LED, then wait and then turn the Onboard LED off. In Program 2, we added a for-loop to reduce redundancy.

Venturing Out - Second Steps - Peripheral Devices

Design Specification

In this part of the lab, we were tasked to write similar programs to that of Program 1 and 2. But, instead of controlling the Onboard LED, we had to control the peripheral LED Bricks. Program 3 had the same blink pattern as Program 1 and Program 4 had the same as Program 2.

Hardware Implementation

To connect the LED Brick to the Arduino, we connected the Electronic Brick chassis onto the top of the Arduino and then connected the LED Brick to the chassis through the D8 connector. All design schematics are located in the appendix.

Software Implementation

The software for these two programs are almost identical to program 1 and 2. The only difference is that we had to set the pinMode in the setup function and then write to the D8 pin instead of the Onboard LED.

Venturing Out - Third Steps - Peripheral Devices

Design Specification

The objective of program 5 was to correctly wire and connect the LED Brick and Button Brick to the arduino. We also had to be able to control the LED with the button, so when the button is pressed, the LED light turns on and when the button is released the LED turns off.

Hardware Implementation

Again, the Electronic Brick chassis is connected on top of the arduino. The LED Brick is connected to D8, and the Button Brick is connected to D9 on the chassis.

Software Implementation

In the setup function, both pinModes for the LED and Button are set to the correct pins with the LED set as an output and the button as an input. Within the loop function, the LED Brick is on or off depending on if the button is pressed or not using the built in digitalRead and digitalWrite functions.

Venturing Out - Final Steps - Processor Outputting Your Stuff

Design Specification

The goal of program 7 was to implement problem 1 from our homework in the Arduino IDE. We are to simply to use the Arduino UNO to display the information regarding the cost of the trip in the serial monitor.

Hardware Implementation

Since there were no peripherals involved the only hardware and connections that we used were the Arduino Uno and the USB connection to a computer.

Software Implementation

The software for this program is directly taken from problem 1 of the homework, as specified in the program specifications. The output regarding the cost of the trip is done in a loop so that the total cost of the trip is displayed in the serial monitor once per second.

Writing to the LCD

Design Specification

In this part of the lab, we introduce the LCD as a peripheral. The goal is to first run the sample program that displays “Hello World” on the LCD. The next step is to modify the text to display the names of the group members on the LCD.

Hardware Implementation

First we install the bus shield to the Arduino UNO. Then we use the 10-pin ribbon cable to connect the LCD to the Bus 1 connector on the bus shield.

Software Implementation

We must first setup the correct pins corresponding to bus 1 on the bus shield that the LCD is connected to. Then we can set up our LCD by specifying its dimensions. Our outputs can be displayed on the LCD through print commands. For the second part of this program printing all of the group members names without overlapping can be done with an additional setCursor function.

Working with the LCD and Peripherals

Design Specification

This program uses two push buttons to manipulate the output on the LCD. First we must display a single letter "X" on the LCD. Then we must enable button1 and button2 to move the "X" left and right across the LCD respectively.

Hardware Implementation

In addition the the LCD connection to the Arduino UNO through the bus shield, we connect two push buttons to ports nine and ten on the bus shield.

Software Implementation

Like the previous program, the correct pins associated with bus 1 must be established for the LCD. In the setup function, we specify the dimensions of the LCD and ports for the buttons. The original "X" is also printed on the LCD. In the loop function, we read the inputs of the push buttons and detect a single move command on every down press of a button. If the "X" reaches the edge of the LCD, no more movements off the edge will be allowed. Also if both buttons are pushed at the same time, there is no movement in the "X."

Communicating Between Two Microprocessors

Design Specification

The last two programs introduce the concept of a local area network. We communicate between two Arduino UNOs through a two wire interface bus. The first program is provided to us and it simply sends information from one Arduino to the other which can be seen on the serial monitors. The second program sends a blinking signal from one Arduino to the other to control both on board and off board LED.

Hardware Implementation

To connect the Arduino UNOs, we must take two wires and connect the respective A4 and A5 pins on each Arduino. Then we connect each Arduino to the PC using the standard USB as we have for all previous programs. To control the LED, we must take the three pins of the LED and connect them to a digital pin, ground, and +5 volts.

Software Implementation

The software implementation of this simple network requires two instances of the Arduino IDE. One for the master and one for the slave. For the first program, we get the master and slave programs from the lab documentation. For the second program, the loop in the master program sends one high and one low signal per cycle. The slave program receives this and directly uses this signal to run the LED, giving it the blinking effect.

ANALYSIS OF ANY ERRORS

Grounding is an issue that often gets overlooked while working with electrical components. In local area networks, data is sent through different devices based on high and low voltage signals. High voltage signal represents 1 in binary while low voltage corresponds to 0 in binary. However, when such high and low voltage signals are sent to a device, the device comprehends those signals relative to its own ground, which means for devices on different groundings, one device's high voltage signal might be interpreted as a low voltage signal to another device on different grounding.

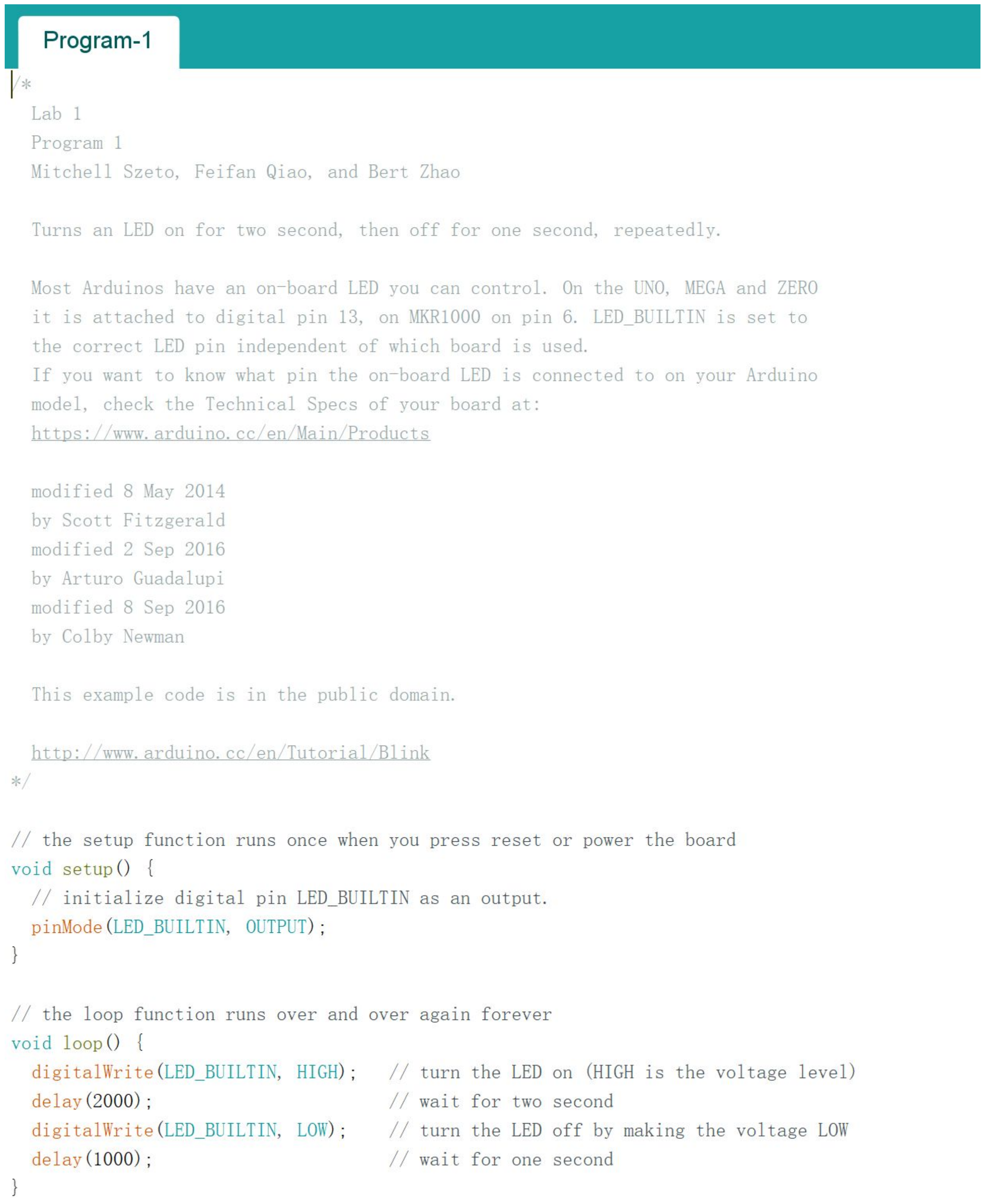
That is what happened when we tried to have two Arduino boards communicate with each other while plugged into two different computers. Each computer, in this case, has its own grounding, so when the slave Arduino board tries to comprehend the incoming voltage levels from the master device, it thinks that the signal is always a low voltage and thus no bits is going through and getting displayed in serial monitor. Once both Arduinos are connected to the same computer, they managed to have a successful communication.

SUMMARY AND CONCLUSION

In this lab, were introduced to the C language and microprocessor programing utilizing both the Arduino IDE and Arduino UNO microprocessor. We used multiple sample programs as starting points for our own programs. We completed 11 basic programs to help introduce us to the concept of embedded programming with the Arduino UNO and the appropriate peripherals. In addition, we also explored basic networking through the connection of two Arduino UNOs. We were able to send some simple information between two Arduino UNOs and display that information through the serial monitor and external LEDs.

Through the completion of this lab, we have successfully completed an introduction to the C language and the Arduino IDE. This lab highlights the foundation for embedded programming by using the Arduino UNO microcontroller. We will build on what we have learned in this lab as we complete projects in future labs.

APPENDICES



```
/*
  Lab 1
  Program 1
  Mitchell Szeto, Feifan Qiao, and Bert Zhao

  Turns an LED on for two second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(2000);                       // wait for two second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for one second
}
```

Program-2

```
/*
  Lab 1
  Program 2
  Mitchell Szeto, Feifan Qiao, Bert Zhao

  Turns the built in LED on for two 1 second blinks, with a 1 second delay between the two blinks
  then turns the LED on for two 2 second blinks, with a 2 second delay between the two blinks

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  for (int i = 0; i < 2; i++) {
    digitalWrite(LED_BUILTIN, HIGH);  // turns the LED on
    delay(1000);                      // delays for one second
    digitalWrite(LED_BUILTIN, LOW);   // turns the LED off
    delay(1000);                      // delays for one second
  }

  for (int i = 0; i < 2; i++) {
    digitalWrite(LED_BUILTIN, HIGH);  // turns the LED on
    delay(2000);                      // delays for two seconds
    digitalWrite(LED_BUILTIN, LOW);   // turns the LED off
    delay(2000);                      // delays for two seconds
  }
}
```

Program-3

```
/*
  Lab 1
  Program 3
  Mitchell Szeto, Feifan Qiao, Bert Zhao

  Turns the external LED Brick on for two second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

int LED = 8; // define the 8th digital pin for LED brick

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED, HIGH);           // turn the LED on (HIGH is the voltage level)
  delay(2000);                       // wait for two second
  digitalWrite(LED, LOW);            // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for one second
}
```

Program-4

```
/*
  Lab 1
  Program 4
  Mitchell Szeto, Feifan Qiao, Bert Zhao

  Turns the built in LED on for two 1 second blinks, with a 1 second delay between the two blinks
  then turns the LED on for two 2 second blinks, with a 2 second delay between the two blinks

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

int LED = 8; // define the 8th digital pin for the LED brick

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  for (int i = 0; i < 2; i++) {
    digitalWrite(LED, HIGH);          // turns the LED on
    delay(1000);                      // delays for one second
    digitalWrite(LED, LOW);           // turns the LED off
    delay(1000);                      // delays for one second
  }

  for (int i = 0; i < 2; i++) {
    digitalWrite(LED, HIGH);          // turns the LED on
    delay(2000);                      // delays for two seconds
    digitalWrite(LED, LOW);           // turns the LED off
    delay(2000);                      // delays for two seconds
  }
}
```

Program-5

```
/*
 * Lab 1 Program 5
 * Mitchell Szeto, Bert Zhao, Feifan Qiao
 *
 * Turns the LED on when the button is pressed
 */


int Button = 9;             //define the 9th digital pin for button brick
int LED = 8;                //define the 8th digital pin for LED brick

void setup() {
  pinMode(LED,OUTPUT);      //set the LED pin for digital output
  pinMode(Button,INPUT);    //set the Button pin for digital input
}

void loop() {
  if (digitalRead(Button))  // if button press
  digitalWrite(LED,HIGH);   // light the LED
  else                      // if not press
  digitalWrite(LED,LOW);    // turn off the LED
}
```

Program-7

```
/*
  Program 7
  Mitchell Szeto, Feifan Qiao, Bert Zhao

  This program calculates the total cost of trip for a given amount of people
  and prints the final cost in serial monitor

  Serial Call and Response in ASCII
 Language: Wiring/Arduino

 This program sends an ASCII A (byte of value 65) on startup
 and repeats that until it gets some data in.
 Then it waits for a byte in the serial port, and
 sends three ASCII-encoded, comma-separated sensor values,
 truncated by a linefeed and carriage return,
 whenever it gets a byte in.

 Thanks to Greg Shakar and Scott Fitzgerald for the improvements

  The circuit:
 * potentiometers attached to analog inputs 0 and 1
 * pushbutton attached to digital I/O 2



 Created 26 Sept. 2005
 by Tom Igoe
 modified 24 Apr 2012
 by Tom Igoe and Scott Fitzgerald

 This example code is in the public domain.
 http://www.arduino.cc/en/Tutorial/SerialCallResponseASCII

 */

 /*
   modified 1 april 2013 by jkp
 */

int people = 9;          // number of people on the trip
float cost = 977.50;     // cost of trip per person
float discount = 0.05;  // discount in the form of a percentage converted to a decimal
float tax = 0.095;       // sales tax in the form of a percentage converted to a decimal


void setup()
{
  // start serial port at 9600 bps and wait for port to open:
  Serial.begin(9600);
}

void loop()
{
  float netCost = cost * (1 - discount) * (1 + tax);

  Serial.print("The total cost of the trip is $");
  Serial.println(netCost * people);
  delay(1000);
}
```

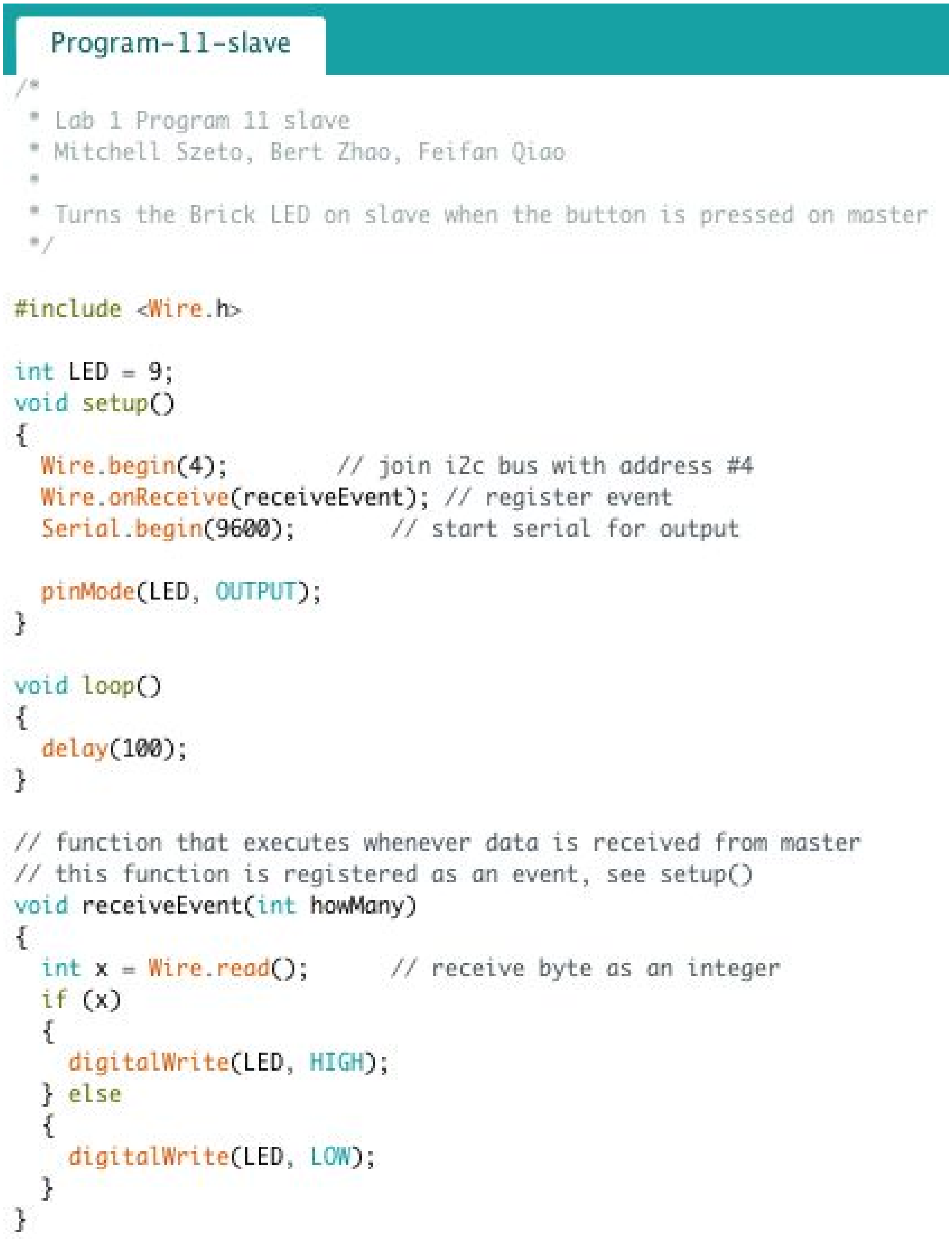
Program-8

```
 modified 7 Nov 2016
 by Arturo Guadalupi

 This example code is in the public domain.

 http://www.arduino.cc/en/Tutorial/LiquidCrystalDisplay

*/

// include the library code:
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
LiquidCrystal lcd(2, 3, 4, 5, 6, 7, 8); // bus 1

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("Mitchell, Feifan");
  lcd.setCursor(0, 1);
  lcd.print("Bert Zhao");
}

void loop() {
  // Turn off the display:
  lcd.noDisplay();
  delay(500);
  // Turn on the display:
  lcd.display();
  delay(500);
}
```

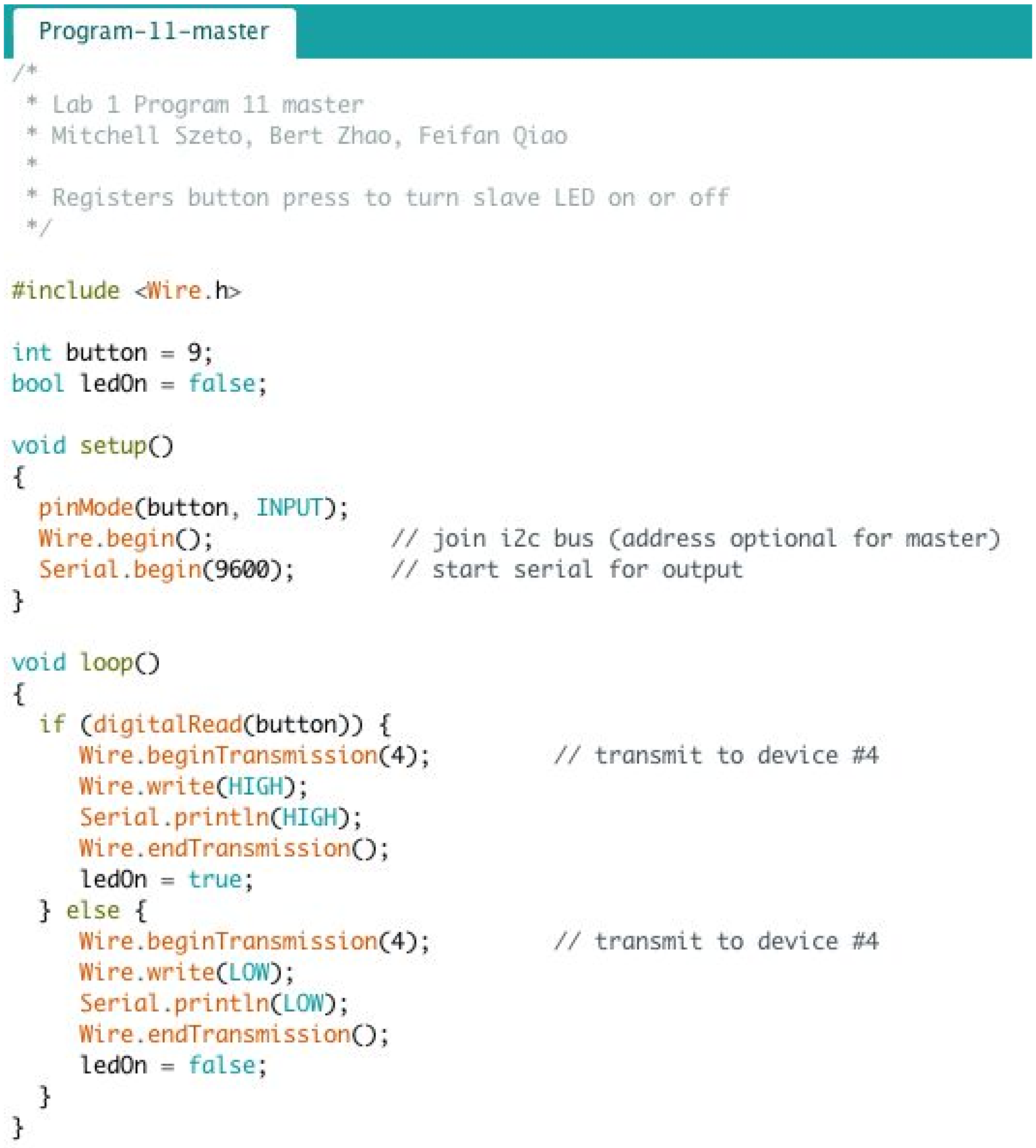
Program-9

```
*/

// include the library code:
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
LiquidCrystal lcd(2, 3, 4, 5, 6, 7, 8);

const int minColumn = 0, maxColumns = 16;
int button1 = 10, button2 = 9;
int lastButton1State = LOW, lastButton2State = LOW;
int column = 0;

void setup() {
  Serial.begin(9600);
  pinMode(button1, INPUT);
  pinMode(button2, INPUT);
  // set up the LCD's number of columns and rows:
  lcd.begin(maxColumns, 2);
  // Print a message to the LCD.
  lcd.print("X");
}

void loop() {
  int read1 = digitalRead(button1);
  int read2 = digitalRead(button2);
  Serial.println(read2);
  if (read1 == HIGH && read1 != lastButton1State) {
    lastButton1State = read1;
    if (column < maxColumns - 1) {
      lcd.scrollDisplayRight();
      column++;
    }
  } else if (read1 == LOW) {
    lastButton1State = read1;
  }

  if (read2 == HIGH && read2 != lastButton2State) {
    lastButton2State = read2;
    if (column > minColumn) {
      lcd.scrollDisplayLeft();
      column--;
    }
  } else if (read2 == LOW) {
    lastButton2State = read2;
  }
```

```
}
```


Program-11-slave

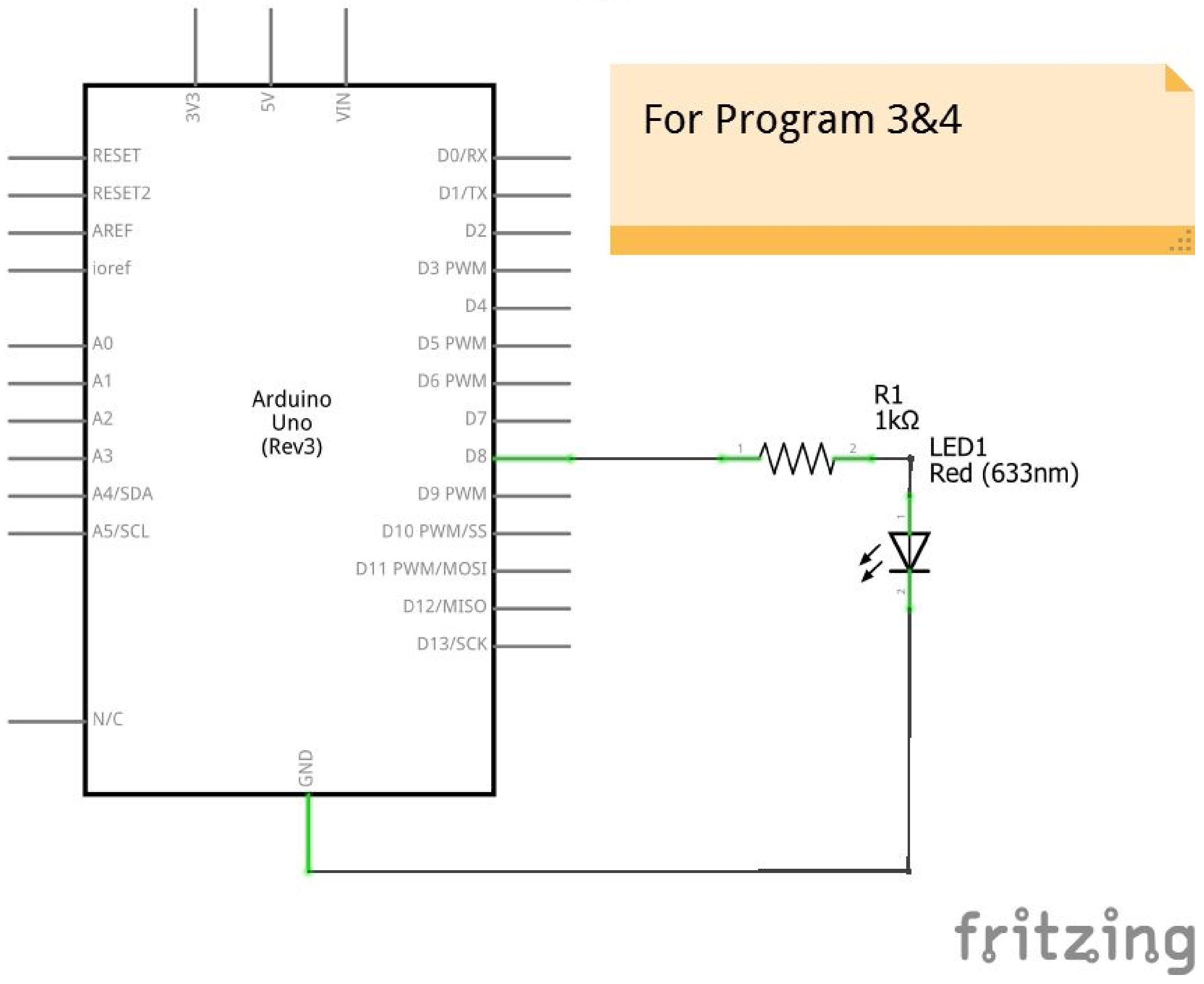
```
/*
 * Lab 1 Program 11 slave
 * Mitchell Szeto, Bert Zhao, Feifan Qiao
 *
 * Turns the Brick LED on slave when the button is pressed on master
 */

#include <Wire.h>

int LED = 9;
void setup()
{
  Wire.begin(4);        // join i2c bus with address #4
  Wire.onReceive(receiveEvent); // register event
  Serial.begin(9600);       // start serial for output

  pinMode(LED, OUTPUT);
}

void loop()
{
  delay(100);
}

// function that executes whenever data is received from master
// this function is registered as an event, see setup()
void receiveEvent(int howMany)
{
  int x = Wire.read();      // receive byte as an integer
  if (x)
  {
    digitalWrite(LED, HIGH);
  } else
  {
    digitalWrite(LED, LOW);
  }
}
```

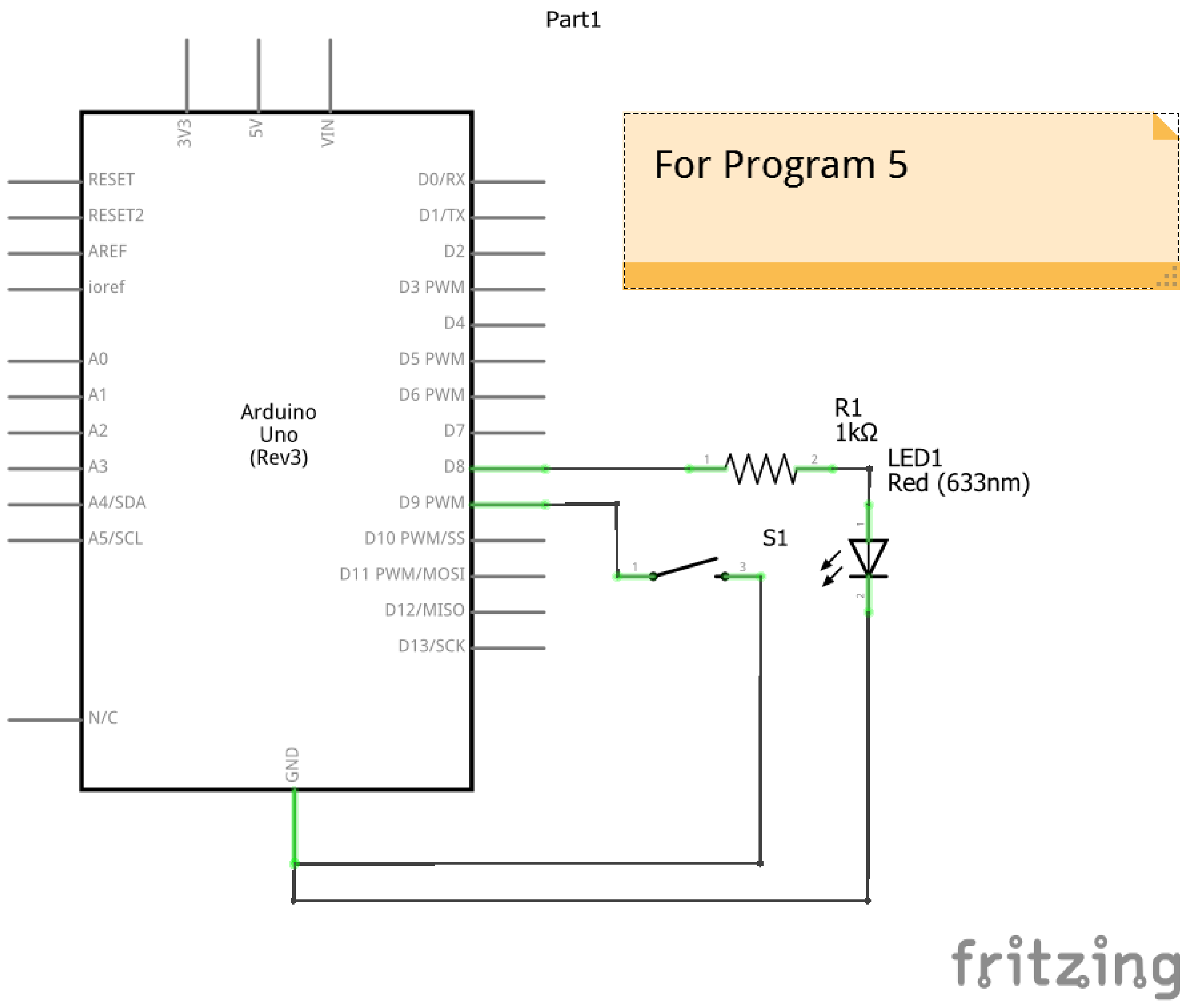


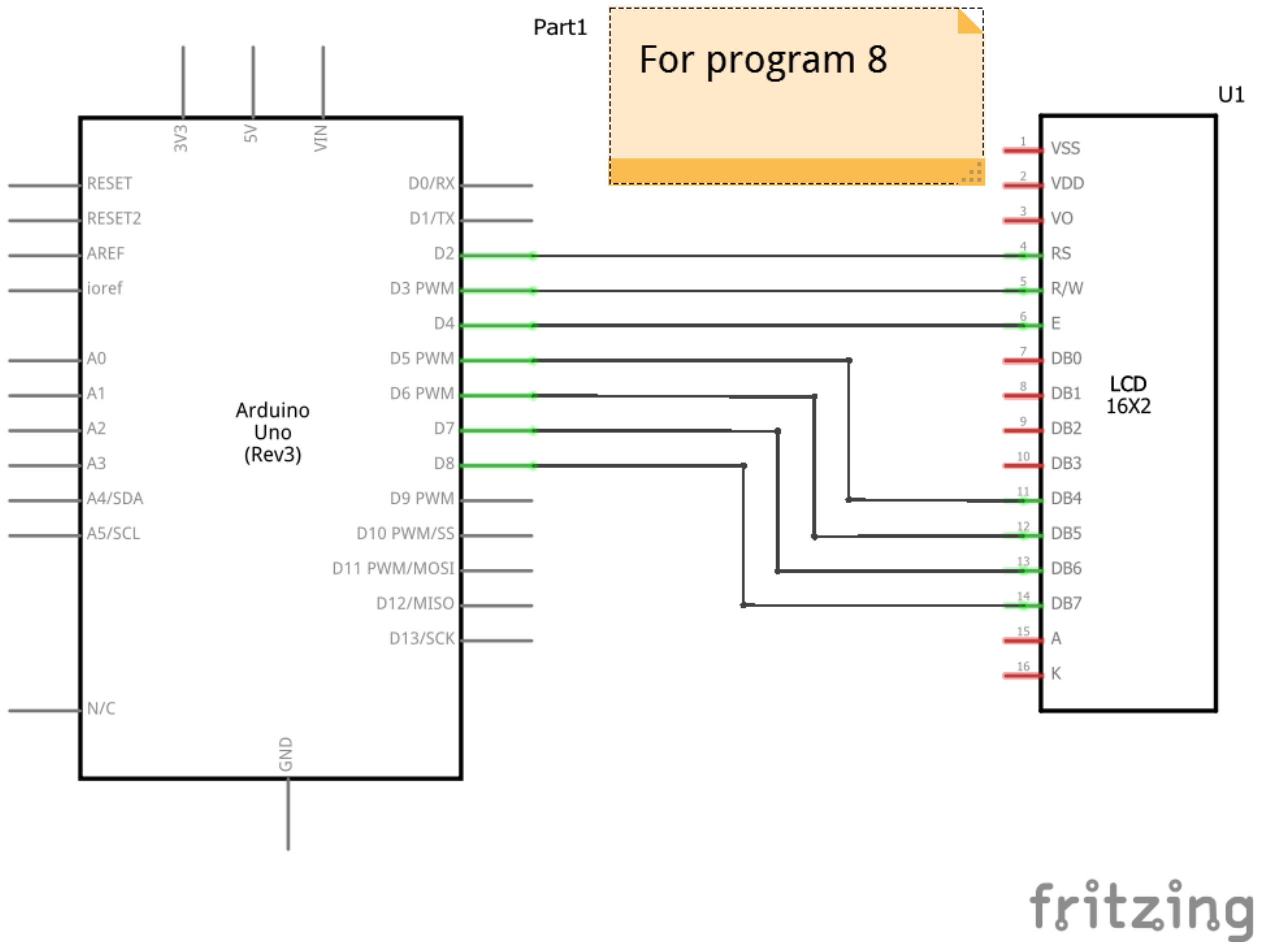
Program-11-master

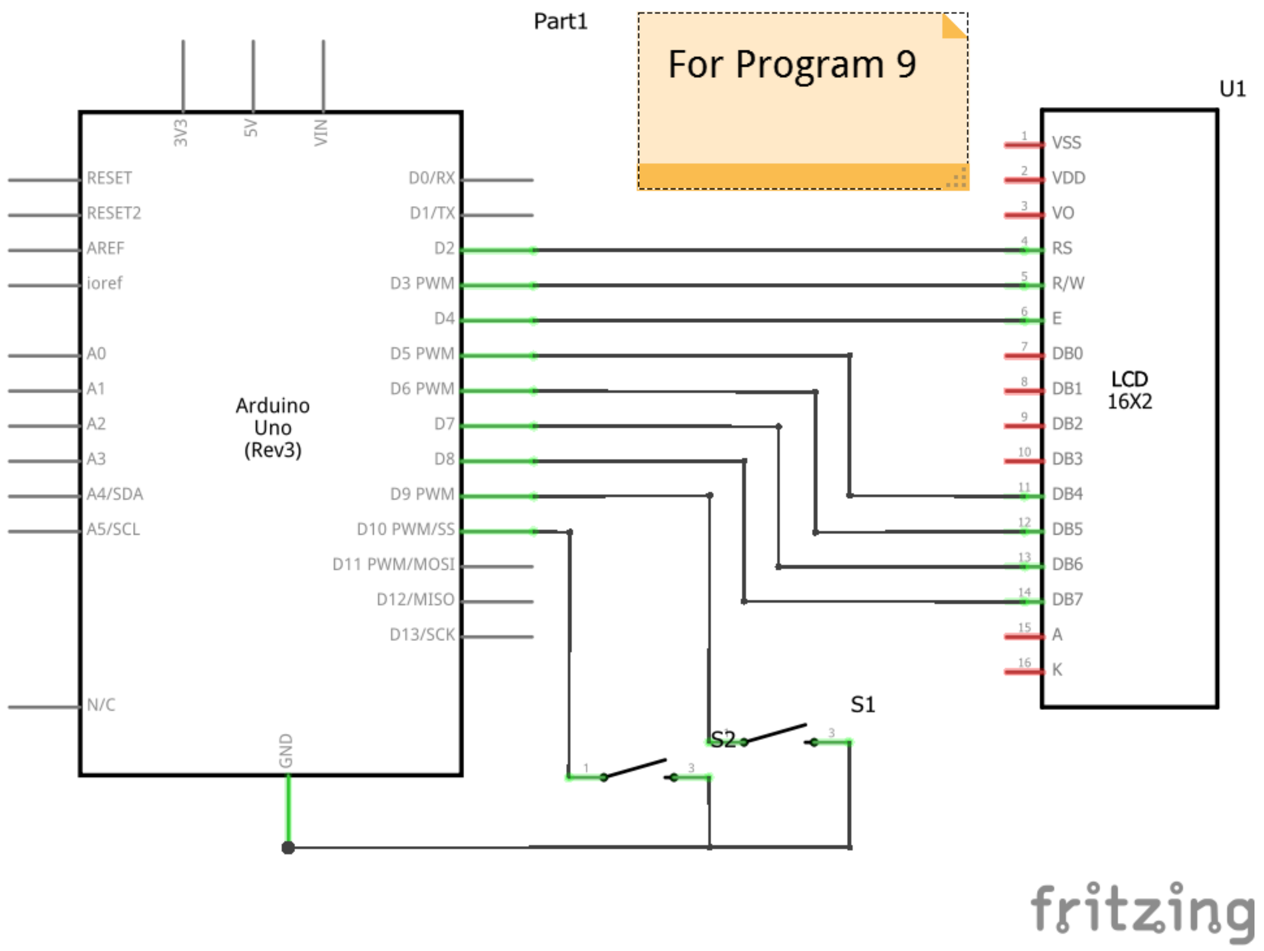
```
/*
 * Lab 1 Program 11 master
 * Mitchell Szeto, Bert Zhao, Feifan Qiao
 *
 * Registers button press to turn slave LED on or off
 */

#include <Wire.h>

int button = 9;
bool ledOn = false;

void setup()
{
  pinMode(button, INPUT);
  Wire.begin();              // join i2c bus (address optional for master)
  Serial.begin(9600);        // start serial for output
}

void loop()
{
  if (digitalRead(button)) {
     Wire.beginTransmission(4);         // transmit to device #4
     Wire.write(HIGH);
     Serial.println(HIGH);
     Wire.endTransmission();
     ledOn = true;
  } else {
     Wire.beginTransmission(4);         // transmit to device #4
     Wire.write(LOW);
     Serial.println(LOW);
     Wire.endTransmission();
     ledOn = false;
  }
}
```

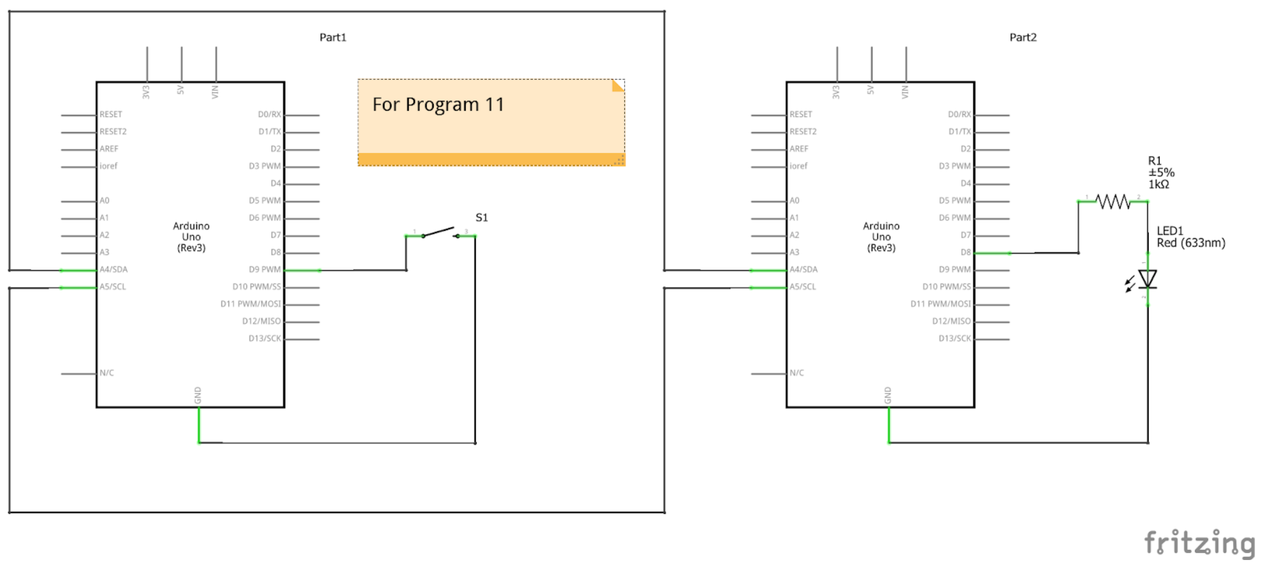
Part1


For Program 3&4
3V3
5V
VIN
RESET
RESET2
AREF
ioref
A0
A1
A2
A3
A4/SDA
A5/SCL
N/C
GND
Arduino Uno (Rev3)
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
R1
1kΩ
LED1
Red (633nm)
fritzing


Part1
For Program 5
3V3
5V
VIN
RESET
RESET2
AREF
ioref
A0
A1
A2
A3
A4/SDA
A5/SCL
N/C
GND
Arduino Uno (Rev3)
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
R1
1kΩ
LED1
Red (633nm)
S1
fritzing


Part1
For program 8
U1
3V3
5V
VIN
RESET
RESET2
AREF
ioref
A0
A1
A2
A3
A4/SDA
A5/SCL
N/C
GND
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
Arduino
Uno
(Rev3)
VSS
VDD
VO
RS
R/W
E
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
A
K
LCD
16X2
fritzing


Part1
For Program 9
U1
Arduino Uno (Rev3)
LCD 16X2
S1
S2
fritzing


Part1
For Program 11
Arduino Uno (Rev3)
S1
Part2
R1 ±5% 1kΩ
LED1 Red (633nm)
fritzing