Marist College

Computer Organization and Architecture

CMPT 422 Final Project

Arduino UNO SMS Door Bell

Author:
Brian Henderson

Professor:
Dr. Pablo Rivas

November 30, 2017

Abstract

The Arduino UNO SMS Door Bell uses an Arduino UNO, and a connection to computer with Internet connection, to send an SMS text message to the door bell recipient. The inspiration for this project is simple: there is no doorbell in my house. The project uses an LCD display, multiple push buttons and resistors, and a potentiometer, to capture user input and display output to the user. Sending a text is done using a connection between the Arduino UNO board and a computer that is capable of executing a python script that utilizes the Twilio API. The user has the option to toggle through a list of names, the names of the residents of the house, and then "confirm," which sends the displayed name a text notifying them that a person has requested them at the door. This helps solve the problem of inefficient, anarchic, and insecure door bells.

1 Introduction

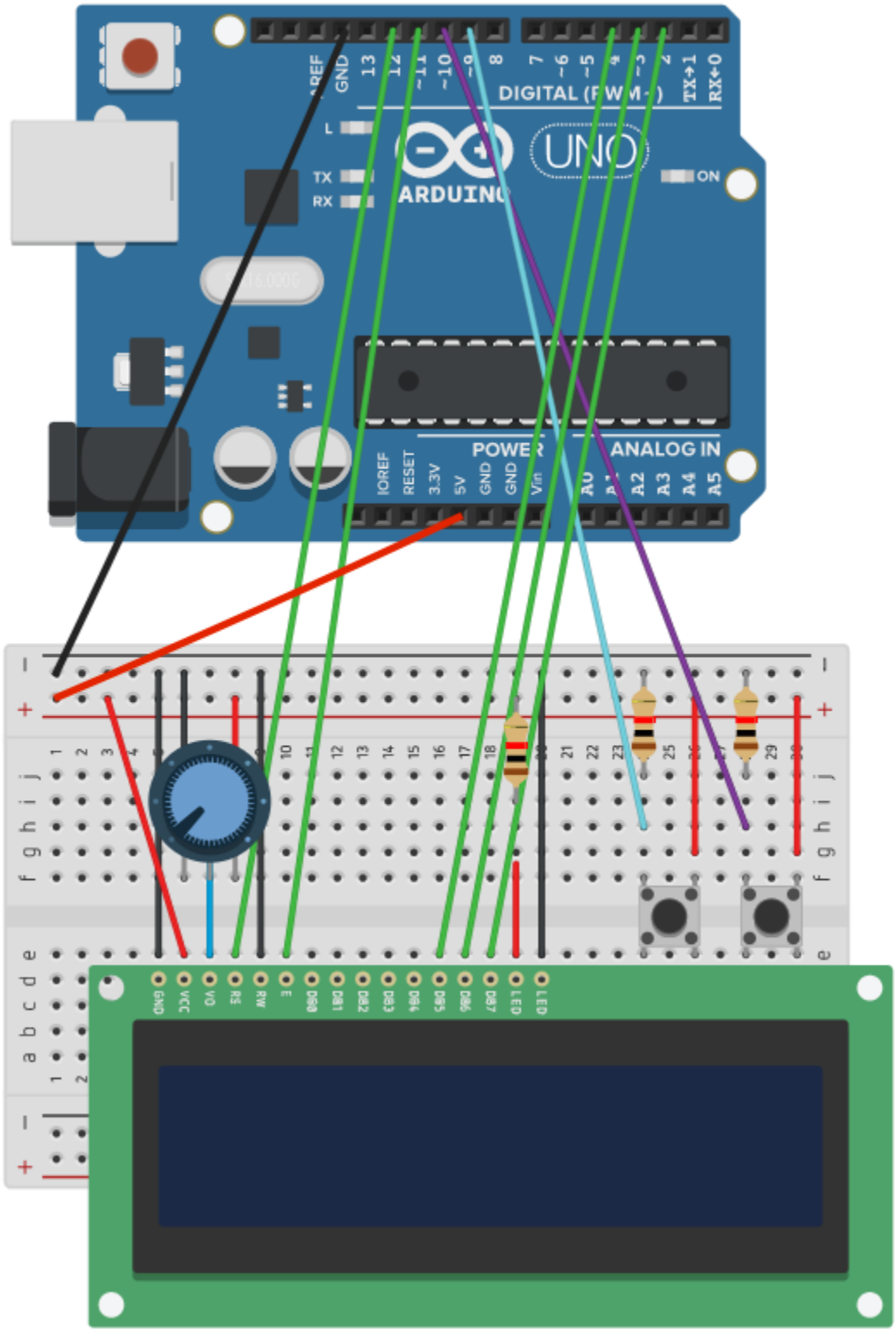
The Arduino UNO SMS Door Bell, an Arduino invention that replaces the traditional doorbell ringing sound, with a SMS text message, customizable to the residents of the household. The inception of this invention came from two factors at my current residence. 1. The doorbell does not work, and 2. If a delivery package requires a signature and nobody answers the door, then a note will be left and the package will be left at a designated pick-up location. This can be a hassle as the pickup location has limited hours, and just one of the inconveniences of not having a doorbell.

2 Methodology

2.1 Description

The Arduino UNO SMS Door Bell uses the following major components: the Arduino UNO board, a 16 pin LCD display, a potentiometer, two push buttons, three 220 OHM resistors, and a computer capable of running a python script with USB interface permissions. The Arduino UNO board is the brain of the system, as it handles the input, output, and computer computations. It receives user input from the two push buttons, and provides output to the LCD screen and the computer terminal. The LCD screen displays a name on the screen, with a button capable of scrolling through all the programmed names. This allows the user to select the appropriate text recipient. Once the user has the desired recipient displayed on the LCD display, the second button can be used to confirm the recipient and send the text. I needed a way for project to send a text message from the push of a button, and in exploring various options, I decided to use the Twilio python API, and pySerial. PySerial is python package that allows the python script to read and write data from a specified serial port. After the send button is pressed, the Arduino UNO prints a message to the usb serial port, which is read from the python script using the pySerial package. Once detected, the python script executes the Twilio code to send a text to the user. The user then receives a standard text, notifying him/her that a person has requested them at the door.

2.2 Schematic



2.3 Arduino Code

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

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int toggleBtnPin = 9;
int confirmBtnPin = 10;
char* contactNames [] = {"Brian Henderson","Zach Toner","Drew Burns","Tanner Senius","Jon Coogan"};
char* contactNumbers[] = {"+17323433511", "+15165927041", "+15185066268", "+18452690469", "+15187286461"};
int currIndex = 0;
int arrSize = sizeof(contactNames)/sizeof(contactNames[0]);


void setup() {
  // set button pins and type
  pinMode(toggleBtnPin, INPUT);
  pinMode(confirmBtnPin, INPUT);
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print(contactNames[currIndex]);
  // Set up serial monitor
  Serial.begin(9600);
}

void loop() {
  // set the cursor to column 0, line 1
  if(digitalRead(toggleBtnPin) == HIGH) {
    lcd.clear();
    currIndex ++;
    if(currIndex == arrSize) {
      currIndex = 0;
    }
    lcd.setCursor(0, 0);
    lcd.print(contactNames[currIndex]);
    delay(200);
  }

  if(digitalRead(confirmBtnPin) == HIGH){
    Serial.println(contactNumbers[currIndex]);
    lcd.setCursor(0, 1);
    lcd.print("Text Sent!");
    delay(200);
  }
}
```

Listing 1: src.ino

The Arduino code is designed to take user input, and output to either the LCD display or the serial port. In order to user the LCD display, the `LiquidCrystal.h` package is used.

The LCD variable takes in six inputs, which are the input buses that the LCD connects to the Arduino UNO with. The `toggleBtnPin` and `confirmBtnPin` are the values of the Arduino inputs from the push buttons. The `contactNames` array is used to store the names of the text recipients, and the `contactNumbers` array is used to store the numbers of the recipients. Note* The number that relates to the name must have the same index in their respected arrays, this was designed this was for simplicity. In the setup function, the pin modes are set, the LCD is established and prints the first name in the array, and the serial monitor opens at a baud rate of 9600. The continuously looping code is pretty straight forward. If the `toggleBtn` receives a high signal (if pressed), the LCD is cleared, the current index goes up one (if it is not the last in the array), and then it prints the next name. The delay makes sure that the button can only scroll through one at a time, as it would otherwise scroll through uncontrollably. If the `confirmBtn` is pressed, then it takes the current index and prints the contact number to the serial, and then displays a confirmation to the LCD display.

2.4 Python Code

```
import serial
from twilio.rest import Client

ser = serial.Serial('COM4', 9600)

account_sid = "AC2c499a37e76bd87fb370afc8d6e57be7"
auth_token  = "####################################"

client = Client(account_sid, auth_token)

def sendText(reciever):
  message = client.messages.create(
    to=reciever,
    from_="+12019480413",
    body="Someone is at the front door for you!")
  print(message.sid)

while True:
  console = ser.readline()
  if console[0] == "+":
    sendText(console)
    print("Text sent")
```

Listing 2: sendSMS.py

The above python script was developed to receive an input from the USB serial port and send a text to that number using the Twilio API. Using the serial library, the variable `ser` is declared to listen on the serial port `'COM4'` at a baud rate of 9600. Every Twilio account credentials have an account SID, and a unique secret token (hashed out in this report), to verify the account. There is one function in this script, `sendText`, which takes one parameter, `receiver`, which is the number to send a text to. The bulk of the functions code is sourced from the Twilio guide, as this is the standard way to send a text. The message contains the

number the text is to be sent to, the number the text derives from, (a number assigned by Twilio), and the message body to be sent. Once the script is executed, the while loop will continuously run and listen on the serial port. All numbers start with a "+", so if a number is pressed, the conditional statement recognizes that and executes the `sendText` function, which sends a text to the appropriate number,

3 Challenges

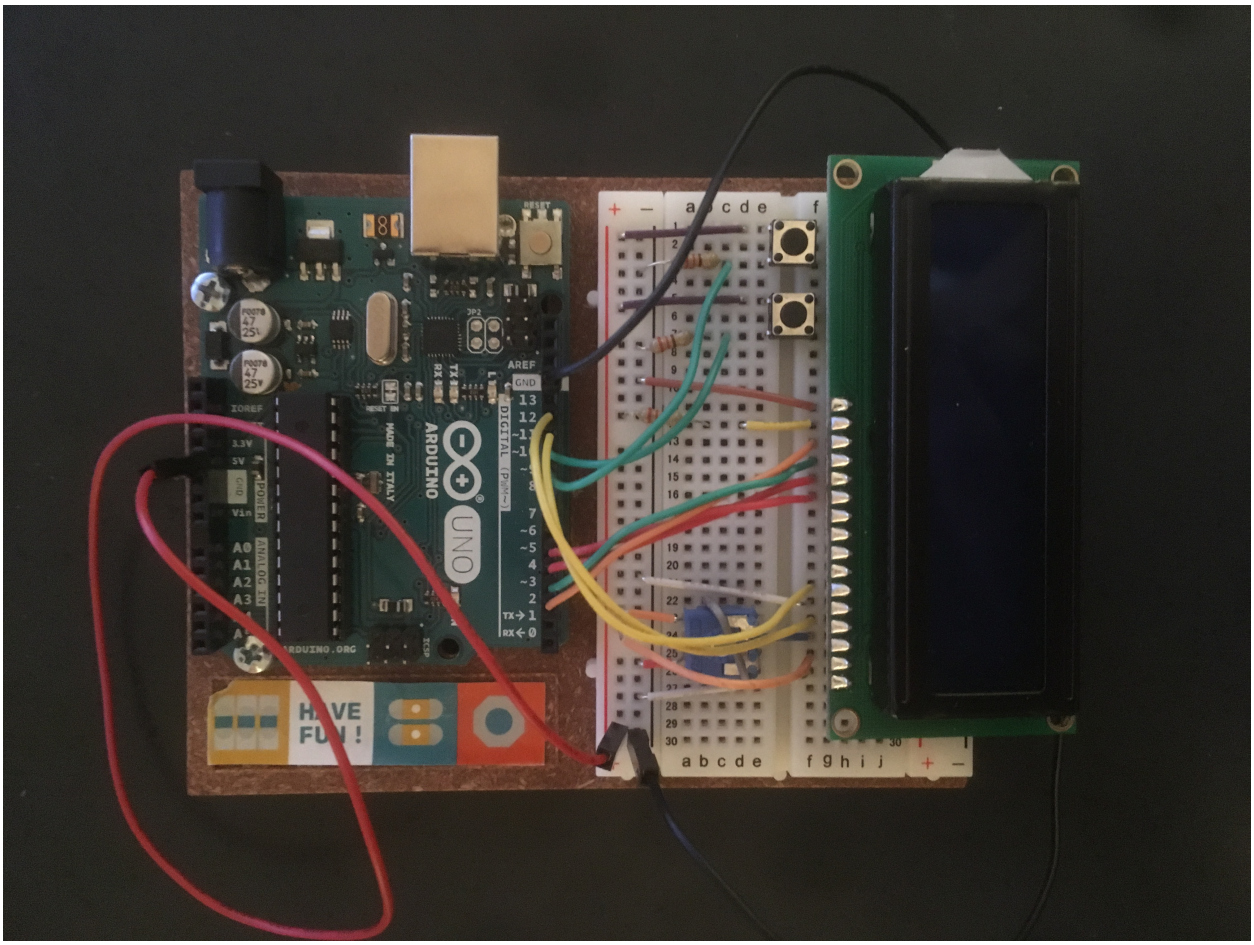
A challenge that I ran into early on was getting the LCD to scroll through the names one by one. I use the `currIndex` variable to keep track of what name is displayed, as the value of `currIndex` was the index value of that name in the array. Once the toggle button is pressed, the value of `currIndex` was supposed to only go up once, but instead it would go up a random amount. I quickly realized, through printing out the value of `currIndex` to the console, that the code block that was to be executed when the toggle button was pressed needed a delay. The delay made sure that the `currIndex` value would only go up by one value, because before the conditional statement would run multiple times in the fraction of a second.

The major challenge that I had faced in the development of this project was getting the project to be able to send a text. Logistically, there were two options: 1) Buy a GSM Shield and program all internally in the Arduino UNO circuit. 2) Use a free trial using the Twilio API and figure out how to connect the Arduino UNO to a computer to execute a script. I decided option two because of price logistics, and to add another challenge, as doing option one would be relatively simple. The challenge that I had ran into was that the pySerial package, the package used to connect a serial port and a python script, was having trouble reading from the USB port. Issues that I ran into were access issues, and trouble identifying the USB port.

4 Coolness

The Arduino UNO SMS Door Bell has a coolness off the charts! Traditional smart doorbells can be in the hundreds of dollars, be a hassle to install, and do not have the customization that comes with an Arduino UNO project. It is also pretty cool because it incorporates a second computer, which may not be the most practical, but it is pretty cool how I was able to get two computers to work cooperatively together.

5 Photo Result



6 References

Twilio, *Communication APIs for SMS, Voice, Video and Authentication*
Available: `https://www.twilio.com/`

S. Fitzgerald and M. Shiloh, *The Arduino Projects Book*. Torino: Arduino, 2012.