1. **Introduction**

Prior to beginning this project, all of the group members came together and decided that we wanted our project to have a meaningful impact on the world. Due to the very prevalent and ongoing pandemic, we wanted to make our project something that helps with the fight against the pandemic. We came up with a project that counts the amount of people entering and exiting a space/room, checks temperature, and sends SMS notifications to notify attendees when a store or other location is ready for them to enter. We decided to create this project because we wanted to optimize the use of employees during these tough times. It is financially wasteful having employees standing by the entry doors all day just to let customers in and out of an area, so we created DigiTalli so we can help business owners through these tough times. The approach we had to this project was dividing the work into smaller teams, and having ongoing communication to ensure that we are constantly integrating all of the components of the project.

2. **Background Research**

<u>Which kit and sensors to order to buy/ order:</u>

Since the focus of this project was to design, implement and present a microcontroller driven project, we knew that we needed to get our hands on a microcontroller. A quick Amazon search gave us plenty of results with one-day shipping but before we ordered anything, we needed to decide which microcontroller to use, specifically whether to choose Arduino or Raspberry Pi. The main difference between them is that Ardiuno is a microcontroller board while the Raspberry Pi is a SBC (microprocessor based mini computer) (Teja, 2021).

Arduino	Raspberry Pi
- Board contains the CPU, RAM and ROM. Additional hardware on the Arduino board is for power, programming and Input/Output connectivity - Does not have any operating system, just need firmware to instruct microcontroller - Good for interfacing sensors, controlling LEDs and motors - Open source hardware and software - Usually used for running a single task repeatedly - Can be programmed using C or C++ - Need additional module or shield to connect to internet - Arduino logic level is 5V and most sensor and modules are designed to be	- Comes with fully functional operating system - Linux is the preferred operating system - Good for developing software application using Python - Can also connect sensors - Power: USB, needs more current than Arduino, power adapter needed, if power cut needs to be restarted - Python IDLE, Eclipse or any text editor that is supported by Linux - Not open sourced - Can perform multiple tasks simultaneously and has a powerful processor and Linux based IOS - Can easily connect to internet or Wi-Fi or Ethernet

compatible	

We considered the pros and cons of each, similarly to the following list below, and ultimately decided to use an Arduino because the programming language is similar to what we were learning in our computer science class and one of our group members had an old kit from home.

Programing/how to code/setting up arduino IDE:

After choosing the microcontroller, we needed to look into how to connect, program and communicate with the board. The Arduino website had excellent documentation on how to get started, the different types of Arduino boards, instructions for shields and how to set up the software. There are two options we found on the Arduino Software Page ("Getting started with Arduino products," 2020):

1. With reliable Internet connection, we can use the online IDE (Arduino Web Editor). It saves sketches on the cloud which automatically backs it up from any device. This ensures that the code is running on the most updated version of the IDE
2. To work offline or from a computer, there is an IDE desktop version.

One of the main reasons we chose Ardiuno was because of how many available resources, tutorials and instructions there are, so learning how the program and board works is relatively straightforward and simple to grasp.

How the sensors and arduino parts and everything in the kit works:

A resource that we referenced frequency to read about the board, certain sensors and how to set everything up was The Most Complete Starter Kit Tutorial document for Arduino Uno (Monk, n.d.). This document explained everything in great detail, including adding libraries, configuring displays, and even the colour chart for the resistors. Our main needed components that we decided to use in the planning process are described below:

The breadboard is used for ease to prototype circuits quickly without having to solder the connections. It is essentially a plastic grid with strips of metal that provide electrical connection between holes in the short middle rows and columns. The stripes on the edge make it convenient to connect power to multiple components and points on the board. They do have some limitations as the connections are push-fit and temporary so they are not as reliable as soldered connections.

Ultrasonic sensors are used in projects that need distance measurements. The ultrasonic sensor has 4 pins, VCC, Trig, Echo and Ground. The HC-SR04 is an inexpensive sensor that provides 2-400cm non-contact measurement functionality with a supposed ranging accuracy that can reach 3mm ("OSEPP HC-SR04," n.d.). Then the module automatically sends eight 40kHz and detects whether there is a pulse signal back. If there is a signal back, the time is captured from sending ultrasonic to returning. A formula is used to calculate the test distance and to convert the

data into units that we are familiar with. It uses the velocity of sound multiplied by the high level time all divided by two.

The difference between an active and passive buzzer is that an active buzzer has a built-in oscillating source, so it will generate sound when electrified. A passive buzzer does not have the same source so it will not make sounds when electrified, instead square waves with frequency between 2k and 5k need to be sent to drive it. The wiring is pretty simple with two pins.

The LED Lights make great indicator lights and use little electricity. Each LED has an anode and cathode and must be wired the correct way. An easy way to tell is the cathode is always the longer leg of the LED. They should be used with a resistor to limit the amount of current flowing through it or it will burn out.

The LCD Display has an LED blacklight and can display two rows with up to 16 characters on each row. The pins on this sensor consist of VSS (ground), VDD (power), VO (adjusts the contrast of LCD), RS (controls where in the LCD's memory we are writing data to) , R/W (read and write pin), E (enabling pin), D0-D7 (more pins that read and write data) and A and K (controls the backlight). This requires the use of the liquid crystal library provided by Ardiuno.

The Keypad (Membrane Switch Module) is a matrix keypad that follows an encoding scheme which allows it to have significantly less output keys than there are pins, so it involves less wires compared to a linear keypad. Similarly, this uses the keypad library provided by Ardiuno.

Sending Text Messages

One of the goals of our project was to have the phone numbers of users stored and used to send them messages. Through our research we discovered there were many ways to send SMS messages from an arduino however they all required a GSM Shield which we didn't have access to and complicated code beyond our skillset. After extensive research we decided to send messages from an excel workbook using Visual Basic and an SMS API (Schalkx, 2015). However, this came with its own set of learning curves.

How to use Visual Basic

First we had to connect the developer tab on excel to start using Visual Basic. We researched how the SMS API worked and integrated it using VBA with the help of a combination of online tutorials and previous knowledge (Meiryo et al., 2013; "SMS sending from .CSV file," n.d.).

How to Connect CSV File to Excel Workbook & Automatically Update

We realized that getting the phone numbers to a CSV file was significantly easier, so we then had to find a way to send the phone numbers to an Excel spreadsheet in a way that the spreadsheet gets updated when the CSV file changes. After doing a lot of research on different methods to connect CSV files with excel spreadsheet and trying out different methods, we found a method that allowed us to get an automatic refresh from the CSV file, but we had to make some

modifications to make the process work for our project, and have the data updated every minute. One drawback for this method was that data could not be updated in periods that are shorter than one minute, however, we decided that this would be sufficient for the purpose of our project.

Contactless Infrared MLX90614 Temperature

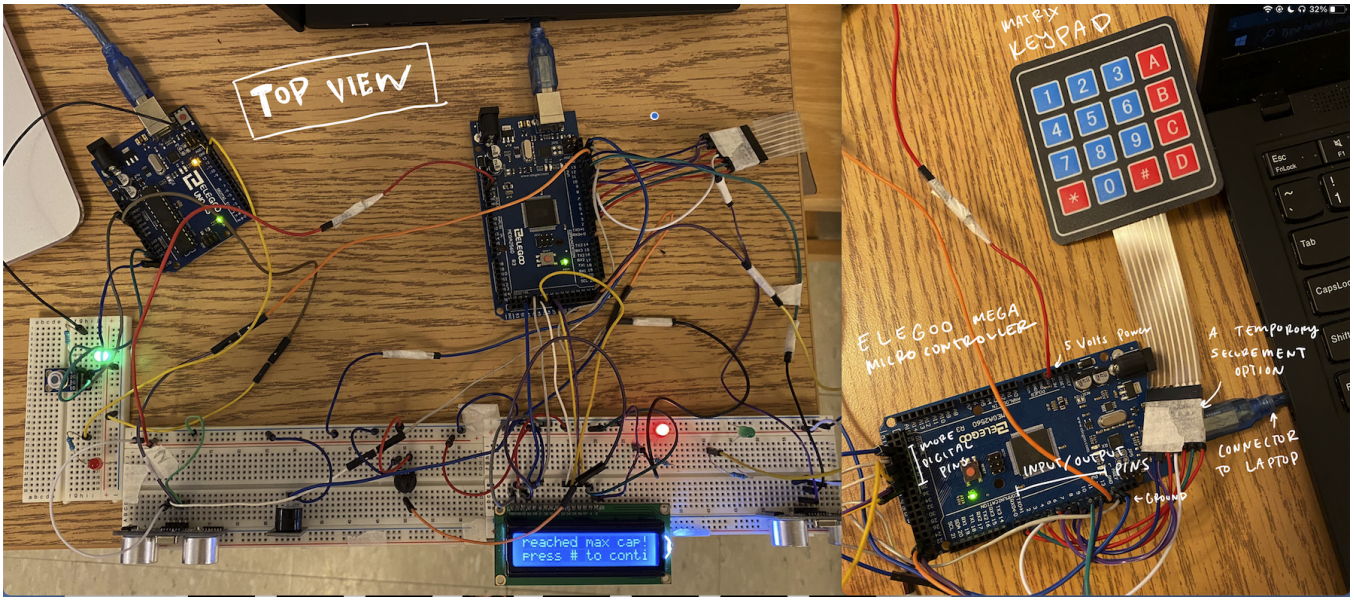
Before implementing our temperature sensor into the project, we had to do some research on exactly how the sensor works. We learned that the contactless sensor works by using IR radiations to measure temperature and communicate this value (hackster.io, n.d.; Raj, 2019). We found that it is able to read both object and ambient (surrounding) temperature (hackster.io, n.d.; Raj, 2019). This is when we decided we would only need to get the object temperature in our code since a person would put their wrist or forehead right in front of the sensor to measure their body temperature. Not only did we need to learn about how the sensor retrieves temperature, but we also spent time researching its hardware. We learned about the four pins on the sensor–Vcc, Gnd, Scl, and Sda–which helped us when it came to wiring the sensor. We discovered that Vcc and Gnd were used to power the sensor, whereas the Scl and Sda were signal pins used for communication (hackster.io, n.d.; Raj, 2019).

1. **Implementation**

Hardware:

Main system:

Our main microcontroller connects everything together. To check the accuracy of each sensor, we placed targets in front of it and recorded each distance. The standard deviation of error is around 9mm which our two sensors fit into the range of. On the breadboard we kept testing the distance between the two sensors, adjusting constants and timing delays in our code until it worked accurately enough to detect the movement.



Although (above) the breadboard wiring is not the most organized, everything is connected and operating correctly. For the LEDs, buzzer, display and keypad, we followed online wiring diagrams and kept track of each corresponding pin number on the microcontroller to later define at the start of the code (below) (Monk, n.d.).

```
#define onboard 13
//define the pins for sensor
#define buzzerPin 25
#define trigPin1 11
#define echoPin1 10
#define trigPin2 13
#define echoPin2 12

#define greenLight 22
#define redLight 23
```

```
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(51, 49, 47, 45, 43, 41);


//connect to rowout pinouts of the keypad
byte pin_rows[ROW_NUM] = {9, 8, 7, 6};

//connect to the column pinouts of the keypad
byte pin_column[COLUMN_NUM] = {5, 4, 3, 2};
```

How we wired everything for temperature sensor:

```
#define redLight 12
#define greenLight 13
```

```
pinMode(redLight, OUTPUT);
pinMode(greenLight, OUTPUT);
```

Our temperature sensor is wired on a different Arduino than our motion sensors and sits right next to it. We followed online wiring diagrams to set this up (Monk, n.d.). We started by putting the sensor into the breadboard, and then connected the pin label with the equivalent arduino pin. For example, we connected ground to the ground pin on the Arduino, SCL to A5, VIN to 5V, and SDA to the A4 pin. We also have two LED lights which tell customers whether or not they can enter. If the green light is on, their temperature is within the range we specified and they can enter. If the red light is on their temperature is too high and they can't enter. To set up the lights we kept track of the pin number on the microcontroller and defined that value in our code.

Software:

Sending Text Messages:

We extracted the phone numbers from the Arduino to a CSV file. First, the user inputs their phone number via the keypad. Using the CoolTerm extension for Arduino, we transmit the data

from the Arduino to a CSV file. We then established a connection from the CSV file to Excel that automatically refreshes every minute, ensuring each phone number inputted is recorded.

The way that we accomplished this was by getting the CSV file date through Get Data, then loading the data to the specific cell in the spreadsheet that we wanted the phone numbers to be copied to. Then we went to Properties and selected Refresh Every Minute, which caused the excel sheet to update every minute and maintain the changes made to the CSV file.

Then we coded a macro in VBA to send an SMS text using the SMS API, sms77.io. The code reads the API key, phone number, and message from their corresponding cells and sends a text to the customer. Each time a message is sent, Excel shifts all the phone numbers stored up one cell, thereby deleting the number from its current cell and storing the next number in that cell, so that the next time a text message is sent, it is sent to the next phone number on the list.

Connecting board to computer and coding the sensors:

Part 1:

```
#include <Arduino.h>
#include <Keypad.h>
#define onboard 13
//define the pins for sensor
#define buzzerPin 35
#define trigPin1 11
#define echoPin1 10
#define trigPin2 13
#define echoPin2 12

int timeoutCounter = 0;
int currentPeople = 0;
//can change the max capacity here
int maxCap = 1;
long duration, distance;
```

Our code begins by including the Arduino and KeyPad library and defining pins that correspond to different pins of our sensor as well as the pin for the buzzer. Next, we initialize variables that we use in later parts of code. These variables are to keep track of the amount of people in a certain space.

Part 2:

```
//four rows
const int ROW_NUM = 4;
//four columns
const int COLUMN_NUM = 4;
char keys[ROW_NUM][COLUMN_NUM] = {
  {'1','2','3', 'A'},
  {'4','5','6', 'B'},
  {'7','8','9', 'C'},
  {'*','0','#', 'D'}
};
//connect to rowout pinouts of the keypad
byte pin_rows[ROW_NUM] = {9, 8, 7, 6};
//connect to the column pinouts of the keypad
byte pin_column[COLUMN_NUM] = {5, 4, 3, 2};
Keypad keypad = Keypad(makeKeymap(keys), pin_rows, pin_column, ROW_NUM, COLUMN_NUM );
String inputPhoneNumber;
long inputInt;
```

In this part of our code, we set up the buttons for our keypad, which customers use to enter their phone numbers. Since we have a matrix keypad rather than a linear keypad, we set the values for each pin similar to a four by four matrix. We then connect our keypad pins to the Arduino pins by defining the values of the pins we used. We also declare a variable of type string where we later store each number the user inputs.

Part 3:

```
//function to measure intial distance
int distanceMeasure(int trigPin, int echoPin){
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  int duration = pulseIn (echoPin, HIGH);
//divde by 2 to get 1 way time
  return (duration/2)/29.1;
}
```

This function, which we later call in a different function, calculates the distance between the sensors and the wall. The function takes in two values, the trig pin number and the echo pin number, which we used to calculate the distance. We set the trig pin on high for 10 second and then set it on low after 10 seconds. This allows us to calculate the amount of time it takes for the ultrasound wave to reflect back and we store this time in a variable called duration. To get our final distance we divide this value by 2 to get the one-way distance and divide the whole value by 29.1, which is the inverse of the speed of sound (in microseconds/centimeter).

Part 4:

```
void setup() {
// maximum number of digit for a number is 10, change if needed
inputPhoneNumber.reserve(10);
//buzzer pin
pinMode(buzzerPin, OUTPUT);
pinMode(trigPin1, OUTPUT);
pinMode(echoPin1, INPUT);
pinMode(trigPin2, OUTPUT);
pinMode(echoPin2, INPUT);
Serial.begin(9600);
  //get the initial distance of wall to sensor
  initialDistance1 = distanceMeasure(trigPin1,echoPin1);
  initialDistance2 = distanceMeasure(trigPin1,echoPin2);
}//end of void setup
```

This is our setup function, which the Arduino calls when a sketch is started and is mainly to initialize variables, pin modes, and do basic set-up. For the pin modes, we tell our program which pins are meant for output and which ones are meant for input. For example, we defined our buzzer as an output, since it won't be taking in anything and will instead be outputting sound. We then calculate the initial distance between our sensors as soon as the sketch starts by calling the distance measure function. We do this because our program works by comparing the distances to the distance between the sensor and the wall to see if it has changed.

To display code on the LCD screen, we used the Liquid Crystal Library and the accompanying functions. We printed and tested most of our code using the serial monitor and as that became more consistent, we simplified it and essentially casted it onto the LCD display. Wiring it correctly did take some time because of the sheer number of pins. But overall, this component of our project was very straightforward.

```
if(distance1 < initialDistance1 && sequence.charAt(0) != '1'){
  sequence += "1";
}else if(distance2 < initialDistance2 && sequence.charAt(0) != '2'){
  sequence += "2";
}
if(sequence.equals("12")){
  currentPeople++;
  sequence="";
  delay(550);
}
else if(sequence.equals("21") && currentPeople > 0){
  currentPeople--;
  sequence="";
  delay(550);
}
  //resets the sequence if it invalid or timesouts
  if(sequence.length() > 2 || sequence.equals("11") || sequence.equals("22") || timeoutCounter > 200){
      sequence = "";
  }
if(sequence.length() == 1){
  timeoutCounter++;
}
else{
  timeoutCounter = 0;
}
```

To track whether someone has entered the room or not, we took advantage of string concatenation and recorded which distance sensor would get triggered first, using 550 millisecond delays to match up the timing of the average motion. Our initial plan to use two ultrasonic sensors was so that we would be able to detect if a person is entering or exiting. By having two sensors, we can figure out which direction the person is walking in through the order in which they are triggered. The variable currentPeople keeps track of how many people are currently in the space and once it has reached the max capacity we set in the start of the code (which can be easily changed for varying businesses and locations), it will trigger a buzzer, switch the led lights from green to red, display full capacity on the LCD screen and initiate the phone number collecting process.

```
void getPhoneNumber(){

  char key = keypad.getKey();
  if (key) {
    if(key != '#'){
      Serial.print(key);
      lcd.print(key);
    }
    else{
      Serial.print(",");

  lcd.setCursor(0,1) ;
  lcd.print("#: ");
  lcd.print(inputPhoneNumber);
  delay(2000);
    }

    if (key >= '0' && key <= '9') {     // only act on numeric keys
      inputPhoneNumber += key;              // append new character to input string
    } else if (key == '#') {
      digitalWrite(buzzerPin, HIGH);
        delay(500);
        digitalWrite(buzzerPin, LOW);
         lcd.clear();
       lcd.print("#: ");
       lcd.print(inputPhoneNumber);
       lcd.clear();

       //clear the input string
       if (inputPhoneNumber.length() > 0) {
        inputInt = inputPhoneNumber.toInt(); // YOU GOT AN INTEGER NUMBER
        inputPhoneNumber = "";               // clear input

      }
    }
  }
```

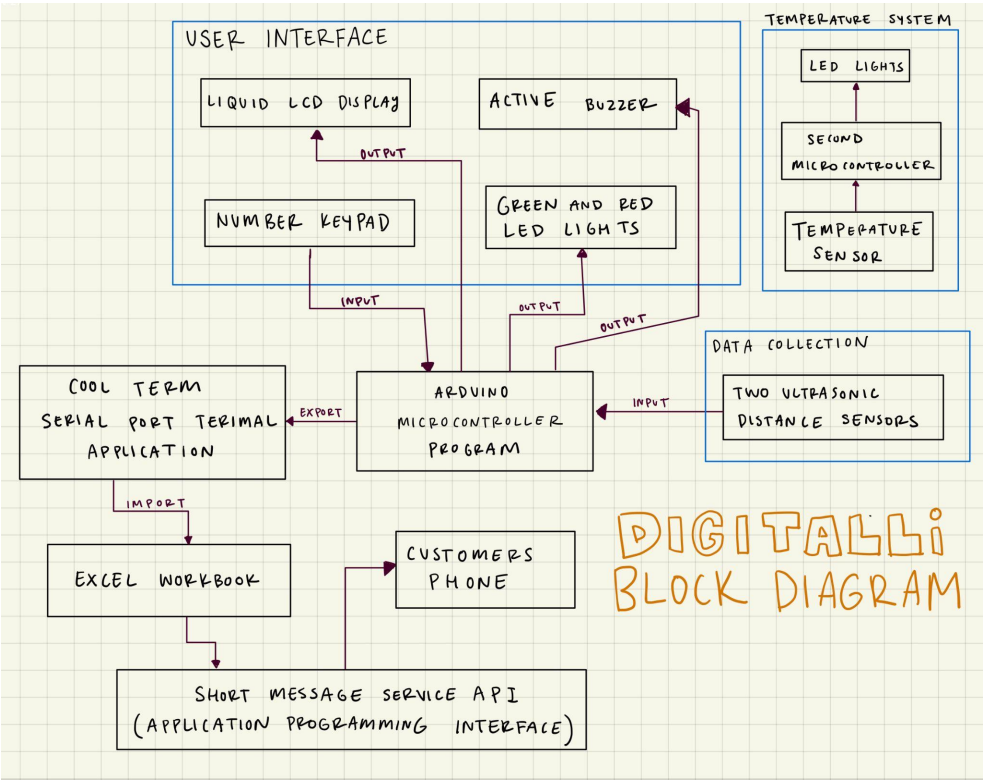
Getting the keypad to work and collect phone numbers was one of the first components of the project that we tackled. Initially, it was in the main loop of our code, but we later turned it into a function that we could call. The function checks if a key is pressed, checks if the key pressed is a valid number, and then proceeds to print it to the LCD screen for the visitor's convenience and to the serial display on the laptop. We used the # symbol on the keypad as an enter key. Once '#' is pressed, it confirms that the phone number is captured, and sends the phone number to later be used with the API.

Temperature sensors code:

For our temperature sensors, we had a few different functions. The getTemp function takes in a char type parameter which represents either Celcius or Farenheit and retrieves the temperature, storing it in the variable called “value”, which it returns at the end. To get the temperature, the functions readObjectTempC() and readObjectTempF() were called depending on if C or F is passed as the parameter. The printTemp function, if called, prints the temperature value, depending on the parameter that indicates type Celcius or Farenheit. The way it does this is that it calls the getTemp function and stores this value in a variable called “ ”. It then prints the value stored in the variable depending on if a C or F was passed as the parameter. The main loop, which keeps running, starts by turning on the green light. This given function helps us control whether or not the light is on or off. We set it as HIGH which means we turn it on at the beginning of the loop. The green light indicates that the temperature is valid. After this, we call the printTemp function for both celsius and fahrenheit by passing C and F as parameters. In the loop function, we have an if statement that checks if the temperature in Celsius is too high which turns on the red light and turns off the green light. If it isn't too high, we turn the green light on and the red off.

Programming in the Arduino IDE was sufficient for this project, but it lacked certain features that most modern code editors have. We followed this tutorial to set up Platform IO which is an open source tool for programming Arduino boards.

Block diagram:



References

Getting started with Arduino products. Arduino. (2020, August 14). Retrieved December 1, 2021, from https://www.arduino.cc/en/Guide.

hackster.io. (n.d.). *Project 014: Arduino MLX90614 Infrared temperature sensor*. Arduino Project Hub. Retrieved December 1, 2021, from https://create.arduino.cc/projecthub/infoelectorials/project-014-arduino-mlx90614-infrared-temperature-sensor-a48bba.

Meiryo, Chuff, & Yosh, m. (2013, January 12). *Open A .CSV file in Excel and have it update whenever .CSV file changes*. Super User. Retrieved December 1, 2021, from https://superuser.com/questions/533018/open-a-csv-file-in-excel-and-have-it-update-whenever-csv-file-changes.

Monk, S. (n.d.). The Most Complete Starter Kit for Uno. Shenzhen; Elegoo Inc.

OSEPP HC-SR04. Allied Electronics & Automation. (n.d.). Retrieved December 1, 2021, from https://ca-en.alliedelec.com/product/osepp/hc-sr04/70669186/.

Raj, A. (2019, July 4). *Make a non-contact infrared thermometer with MLX90614 IR temperature sensor*. Circuit Digest. Retrieved December 1, 2021, from https://circuitdigest.com/microcontroller-projects/ir-thermometer-using-arduino-and-ir-temperature-sensor.

Schalkx, C. (2015, October 13). *[how to] send personalised SMS messages using Excel*. CM.com. Retrieved December 1, 2021, from https://www.cm.com/blog/how-to-sending-personalised-sms-messages-using-excel/.

SMS sending from .CSV file. MailUp User Manual. (n.d.). Retrieved December 1, 2021, from https://help.mailup.com/display/mailupapi/SMS+sending+from+.CSV+file.

Teja, R. (2021, September 8). *What are the differences between Raspberry Pi and Arduino?* Electronics Hub. Retrieved December 1, 2021, from https://www.electronicshub.org/raspberry-pi-vs-arduino/.