

Student name: Hoang Long Tran

Student ID: s223128143

SIT225: Data Capture Technologies

Activity 1.1: Arduino Blink

Welcome to Arduino!

Arduino is an electronic prototyping platform. Different types of sensors & actuators can be attached to Arduino boards to create our own sensing-thinking-acting systems.

Throughout this unit, we will use Arduino to create different sensing devices, and to retrieve the collected sensor data.

In this task, we will try out an introductory exercise, to learn the basic concepts of Arduino.

Hardware Required

- Arduino Board with in-built LED

- USB cable

Software Required

- Arduino programming environment

Pre-requisites: You must do the following before this task

Unit site weekly materials

Active learning sessions require reading material and/or videos available in the unit site, which you must read/view **BEFORE** you start the lab. If you are on-campus, this means that we expect you to have gone through these materials when you join active learning sessions.

Why should you read/watch pre-lab materials?

These materials will help you understand the background which the lab tasks require. Students come to university from diverse backgrounds. Some of you may be familiar with the background information, some of you may not. When you come to the lab prepared, you're already equipped

with confidence and will be able to participate in activities better. Ultimately, class time will be much more productive, dynamic, and fun for everyone.

Here are the pre-lab materials for our first task:

1. Watch TED Talk: <https://www.youtube.com/watch?v=UoBUXOOdLXY> (~15 minutes)
2. Read Arduino tutorial: <https://www.dummies.com/article/technology/computers/hardware/arduino/how-to-complete-your-first-arduino-sketch-164747>)
3. Read this task sheet from beginning to end.

Task Objective

“We have an Arduino board with an in-built LED light. We need the LED light to be turned on and off continuously, every one second.”

Activity Submission Details

Answer the questions below in this word document and other activities in this activity sheet to create a PDF and submit to OnTrack as described in this week’s OnTrack task. PDFs of this activity sheet and OnTrack task need to be merged for submission in the OnTrack portal.

Q1: The TED talk given under the Pre-Lab materials, shows how Arduino is being used for interesting projects to capture data from the environment, process it, and use it to carry out useful actions.

Fill the given table below to answer the following:

What are **three** projects that use captured data as given in the TED talk? What data do they capture? What sensors do you think they could use to capture this data?

Project name	Data captured	Sensors to capture the data
Botanicalls	The moisture data will be the language of the plants. There are thresholds set for the moisture level and each threshold there will be an output (e.g. I need water).	Sensor probe measures the moisture of the soil. A microcontroller captures the data and send them online via an embedded ethernet connection.
RDTN Geiger Shield	Radiation data gather from many different sources to inform people of places that are radioactive in Japan since the Fukushima Daiichi nuclear disaster.	Geiger counter and Arduino integrated to capture radiation data.
Tweets farts chair	Natural gas data, more	A office chair and a natural gas

	specifically, human fart that is naturally produced in the digestive system	sensor to capture every time someone farts on the chair, and tweets the fart on Twitter.
--	---	--

Q2: Consider the given Task Objective. Think about how this simple system can be decomposed to ‘Sense-Think-Act’?

- a) What is the ‘sensing’ requirement in this system, if any?
Sensing comes from sensors, and since there is no sensor involved in this case, so there is no ‘sensing’ requirement.
- b) What is the ‘thinking’ requirement in this system, if any?
The thinking or logic of our system is the ability to turn on and off the LED light continuously at 1 second interval.
- c) What is the ‘acting’ requirement in this system, if any?
The acting is the action of the system, which is turning the LED light on and off at appropriate intervals.

Q3: Please refer to the provided ‘Arduino Blink Activity Sheet’ and follow the steps.

- a) In Arduino-speak, what is a “sketch”?
The term ‘sketch’ in this case comes from the functionality of Arduino that allows people to quickly prototype and test ideas using little code, similar to what we sketch on a paper.
- b) `setup()` and `loop()` are key Arduino constructs. These are required in every Arduino sketch.
 - i) Which of the above two, runs once at the very beginning of your program and never again (unless you reset or upload new code)?
The `setup()` function runs every time the program starts. This is where we set up our initialization tasks.
 - ii) Which of the above two, is used to continuously run code over and over again?
The `loop()` is used to run the code continuously in a loop. This is where the logic of our program lies.
- c) What does **`pinMode()`** do?
Hint: <http://arduino.cc/en/Reference/HomePage>
This is a function where we specified a pin to behave as an input or an output.
- d) What is a comment?
Comment is code documenting. Denote with `//` to tell us what this function or variable or any logic in the code do.
- e) What does the following line of code do:
`delay(x);`
Hint: <http://arduino.cc/en/Reference/HomePage>
Function to pause the program execution for a specified period of time.
- f) There is something you need to check before uploading your sketch. What is this?
When we try to upload the sketch to the Arduino. First, the Arduino IDE will verify your code. This does not guarantee the code will work as expected, but it will

show that there are no syntax errors, and the program can be compiled. After successfully compiled, the code will then be uploaded to Arduino.

Q4: How can you test the Blink program to make sure it is working as given in the Task Objective?

We need to first connect to Arduino Nano 33 to our computer. Then we download the correct board manager, choose the right port. Next, we download the sample code called `Blink` and try to upload that to our Arduino. If the code has been successfully uploaded and the built-in LED light is switching at correct interval, then we can conclude that the program is working correctly according to the task objective.

Activity 1.2: Write Arduino data to serial communication port

Now you can blink Arduino's built-in LED, it is time to talk to outside Arduino-world, your computer which connects the Arduino board using a USB cable. Arduino IDE shows what you write to the serial port.

Hardware Required

Arduino Board with in-built LED

USB cable

Software Required

Arduino programming environment

Steps:

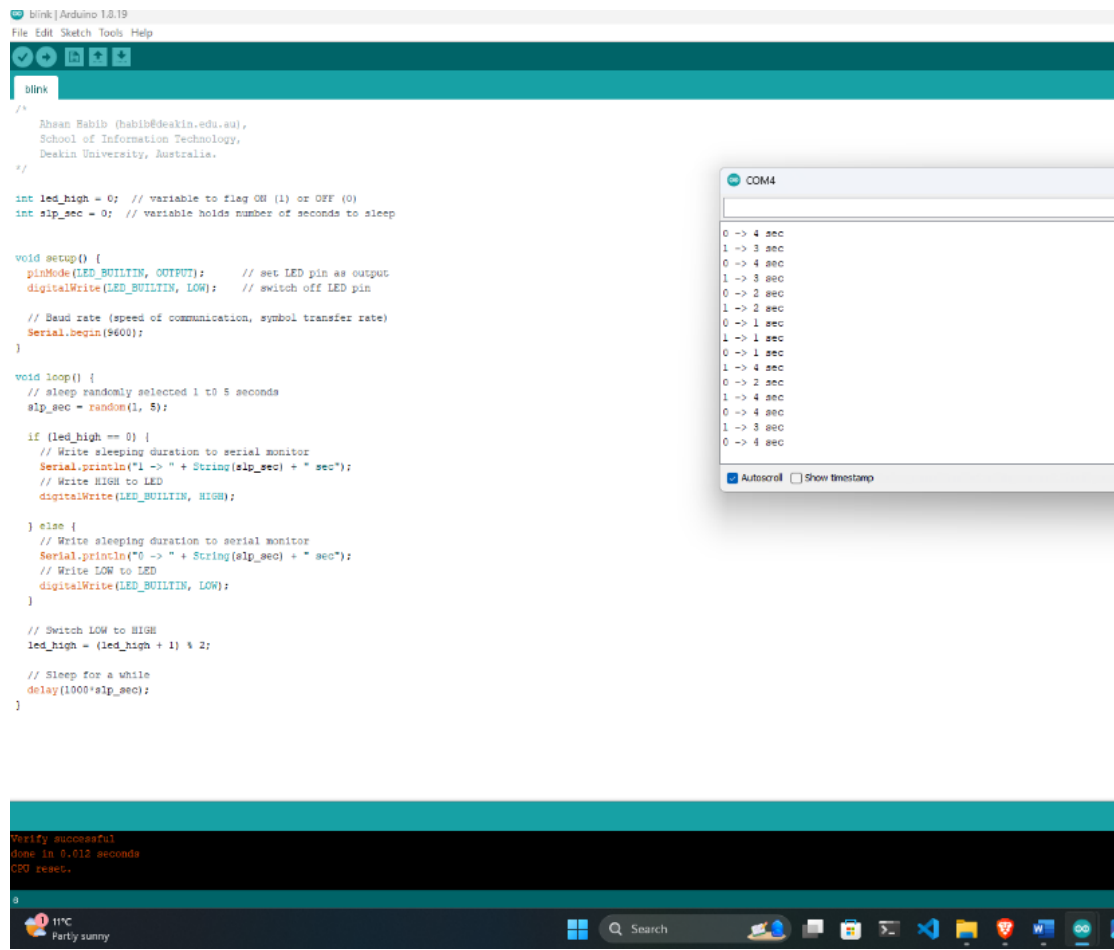
Steps	Actions
1	Identify the port through which Arduino is connected to your computer. You can find it in Arduino IDE Tools menu. Write Arduino sketch (or download from https://github.com/deakin-deep-dreamer/sit225/blob/main/week_1/sketch_blink.ino), which look like below, in Arduino IDE, deploy in your board and observe the output in IDE's serial monitor.

```

1  int led_high = 0; // variable to flag ON (1) or OFF (0)
2  int slp_sec = 0; // variable holds number of seconds to sleep
3
4
5  void setup() {
6      pinMode(LED_BUILTIN, OUTPUT); // set LED pin as output
7      digitalWrite(LED_BUILTIN, LOW); // switch off LED pin
8
9      // Baud rate (speed of communication, symbol transfer rate)
10     Serial.begin(6900);
11 }
12
13 void loop() {
14     // sleep randomly selected 1 to 5 seconds
15     slp_sec = random(1, 5);
16
17     if (led_high == 0) {
18         // Write sleeping duration to serial monitor
19         Serial.println("1 -> " + String(slp_sec) + " sec");
20         // Write HIGH to LED
21         digitalWrite(LED_BUILTIN, HIGH);
22     } else {
23         // Write sleeping duration to serial monitor
24         Serial.println("0 -> " + String(slp_sec) + " sec");
25         // Write LOW to LED
26         digitalWrite(LED_BUILTIN, LOW);
27     }
28
29     // Switch LOW to HIGH
30     led_high = (led_high + 1) % 2;
31
32     // Sleep for a while
33     delay(1000*slp_sec);
34 }
35
36

```

Question: Screenshot the serial monitor output and paste the image here.

	<p>Answer:</p>  <pre> /* Ahsan Habib (ahabib@deakin.edu.au), School of Information Technology, Deakin University, Australia. */ int led_high = 0; // variable to flag ON (1) or OFF (0) int slp_sec = 0; // variable holds number of seconds to sleep void setup() { pinMode(LED_BUILTIN, OUTPUT); // set LED pin as output digitalWrite(LED_BUILTIN, LOW); // switch off LED pin // Baud rate (speed of communication, symbol transfer rate) Serial.begin(9600); } void loop() { // sleep randomly selected 1 to 5 seconds slp_sec = random(1, 5); if (led_high == 0) { // Write sleeping duration to serial monitor Serial.println("1 -> " + String(slp_sec) + " sec"); // Write HIGH to LED digitalWrite(LED_BUILTIN, HIGH); } else { // Write sleeping duration to serial monitor Serial.println("0 -> " + String(slp_sec) + " sec"); // Write LOW to LED digitalWrite(LED_BUILTIN, LOW); } // Switch LOW to HIGH led_high = (led_high + 1) % 2; // Sleep for a while delay(1000*slp_sec); } </pre> <p>Verify successful Done in 0.012 seconds CPU reset.</p>
2	<p>Question: Observe the use of “Serial” such as functions Serial.begin() in setup and Serial.println() in loop. Describe what these functions are doing with respect to the serial monitor output you have attached above.</p> <p>Answer:</p> <p>The `Serial.begin(9600);` sets up the serial data communication at 9600 rate in bits per second (baud). The baud rate of serial monitor needs to match `Serial.begin(9600);` for proper communication. There is a random sleep duration between 1 to 4 seconds to control how long the built in LED will stay on and off. The duration is then printed with `Serial.println`.</p>
3	<p>Question: If Arduino transfers data at 4800 bits per second (baud rate) and you're sending 12 bytes of data, how long does it take to send over this information?</p> <p>Answer: 12 bytes = 8*12 = 96 bits. The speed will be 96/4800 = 0.02 seconds.</p>

Activity 1.3: Arduino talks to Python

To listen to what Arduino sends, there will be a Python program running and keep listening to the same port where Arduino is writing data to receive it.

Hardware Required

Arduino Board with in-built LED

USB cable

Software Required

Arduino programming environment

Python 3.0 (Follow Python installation manual in unit site)

Steps:

Steps	Actions
1	Write Arduino sketch (or download from https://github.com/deakin-deep-dreamer/sit225/blob/main/week_1/sketch_serial_comm.ino) which looks like below. Open in Arduino IDE. Study the code. Upload the code to Arduino board and observe output in Arduino IDE serial monitor.

```

1
2 int x;
3
4 void setup() {
5     Serial.begin(9600); // set baud rate
6 }
7
8 void loop() {
9     while (!Serial.available()) {} // wait for data to arrive
10
11     // read string data from Serial, we know Python
12     // script is sending just an integer.
13     x = Serial.readString().toInt();
14
15     // write a string (no newline)
16     Serial.print("Arduino sends: ");
17
18     // Add 1 to what received.
19     // Write an integer with a newline (println vs print).
20     Serial.println(x + 1);
21
22     // Push the data through serial channel.
23     Serial.flush();
24 }
25

```

Question: Do you see any output in serial monitor? If not, then why?

Answer: There is no output in the serial monitor because there is no python code that sent data to read.

- 2 Write Python code as below (or download from https://github.com/deakin-deep-dreamer/sit225/blob/main/week_1/serial_comm_script.py) and save it to a file serial_comm_script.py.

```
File Edit Selection View ... 1.1
serial_comm_script.py X
activity 1.3 > serial_comm_script > serial_comm_script.py > ...
7 import serial
8 import random
9
10 # set baud rate, same speed as set in your Arduino sketch.
11 boud_rate = 9600
12
13 # set serial port as suits your operating system
14 s = serial.Serial('COM4', boud_rate, timeout=5)
15
16 while True: # infinite loop, keep running
17
18     # a random number between 5 and 50.
19     data_send = random.randint(5, 50)
20
21     # write to serial port, set data encoding.
22     # Raw bytes are sent through serial ports, Python bytes() needs
23     # to know the encoding to generate bytes from string.
24     #
25     # We send a single integer which is read from Arduino sketch.
26     #
27     d = s.write(bytes(str(data_send), 'utf-8'))
28     print(f"Send >>> {data_send} ({d} bytes)")
29
30     # Read from serial port.
31     #
32     # readline keeps reading until a newline found in the data stream.
33     # Unlike write above, we just send an integer with no newline.
34     # You should receive data the same way as it is sent.
35     #
36     d = s.readline().decode("utf-8")
37     print(f"Recv <<< {d}")
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

PS C:\Users\tomde\OneDrive\Documents\Deakin\Deakin-Data-Science\T1Y2\SIT225 - Data Capture Technologies\Week 1 - Introduction to SIT225 Data Capture Technologies\1.1> python -u "c:\Users\tomde\OneDrive\Documents\Deakin\Deakin-Data-Science\T1Y2\SIT225 - Data Capture Technologies\Week 1 - Introduction to SIT225 Data Capture Technologies\1.1\activity 1.3\serial_comm_script\serial_comm_script.py"

Traceback (most recent call last):

File "c:\Users\tomde\OneDrive\Documents\Deakin\Deakin-Data-Science\T1Y2\SIT225 - Data Capture Technologies\Week 1 - Introduction to SIT225 Data Capture Technologies\1.1\activity 1.3\serial_comm_script\serial_comm_script.py", line 14, in <module>

s = serial.Serial('COM4', boud_rate, timeout=5)

File "C:\Users\tomde\AppData\Local\Programs\Python\Python311\Lib\site-packages\serial\serialwin32.py", line 33, in __init__

super(Serial, self).__init__(*args, **kwargs)

File "C:\Users\tomde\AppData\Local\Programs\Python\Python311\Lib\site-packages\serial\serialutil.py", line 244, in __init__

self.open()

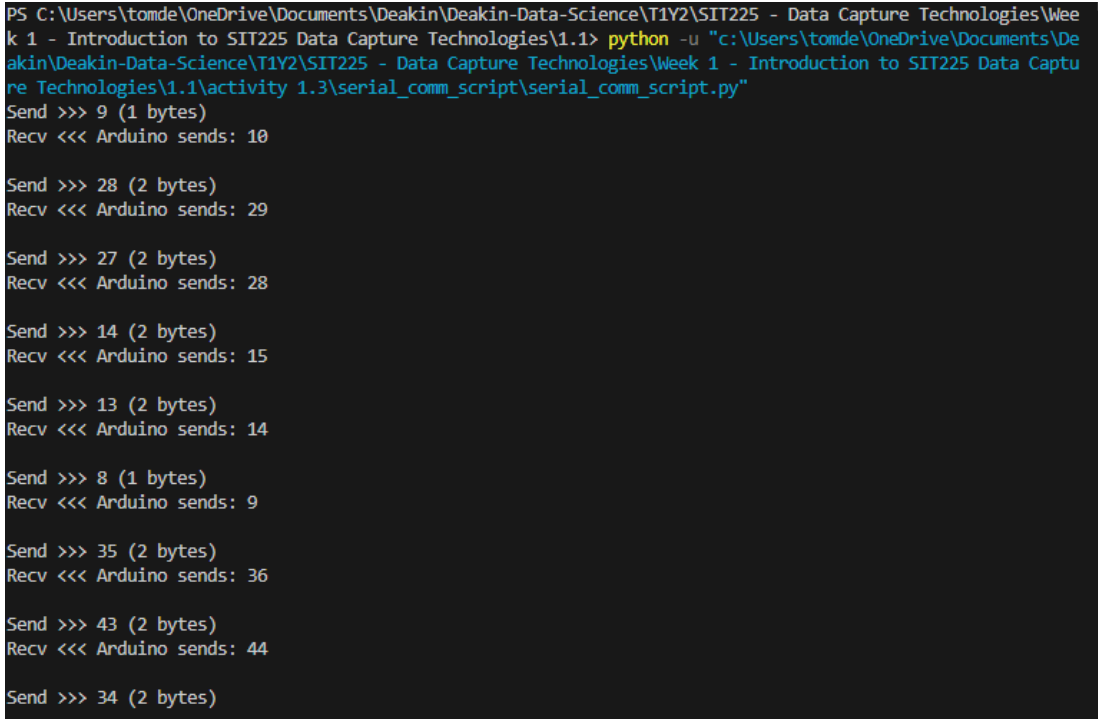
File "C:\Users\tomde\AppData\Local\Programs\Python\Python311\Lib\site-packages\serial\serialwin32.py", line 64, in open

Run the Python file from command line using command:

\$ python serial_comm_script.py

Question: Run Python script in command line, does the script run or do you receive any error? If there is an error, analyse the error message and identify what went wrong.

Answer: The python script when runs receive errors because the Arduino IDE is still running and using the port.

3	<p>Question: Following the above step 2, do you think Arduino IDE is keeping the serial communication port busy talking to Arduino board? Now try to close the Arduino IDE and repeat step 2 above to run the Python script. Do previous errors show up again? If not, Python script should have print messages in the command line. Take the screenshot of the Python script output and paste here.</p> <p>Answer:</p>  <pre> PS C:\Users\tomde\OneDrive\Documents\Deakin\Deakin-Data-Science\T1Y2\SIT225 - Data Capture Technologies\Week 1 - Introduction to SIT225 Data Capture Technologies\1.1> python -u "c:\Users\tomde\OneDrive\Documents\Deakin\Deakin-Data-Science\T1Y2\SIT225 - Data Capture Technologies\Week 1 - Introduction to SIT225 Data Capture Technologies\1.1\activity 1.3\serial_comm_script\serial_comm_script.py" Send >>> 9 (1 bytes) Recv <<< Arduino sends: 10 Send >>> 28 (2 bytes) Recv <<< Arduino sends: 29 Send >>> 27 (2 bytes) Recv <<< Arduino sends: 28 Send >>> 14 (2 bytes) Recv <<< Arduino sends: 15 Send >>> 13 (2 bytes) Recv <<< Arduino sends: 14 Send >>> 8 (1 bytes) Recv <<< Arduino sends: 9 Send >>> 35 (2 bytes) Recv <<< Arduino sends: 36 Send >>> 43 (2 bytes) Recv <<< Arduino sends: 44 Send >>> 34 (2 bytes) </pre>
4	<p>Question: Observe the Python script output and describe the communication protocol used between Arduino sketch and Python script.</p> <p>Answer: Python sends data integer type randomly between 5 to 50. Arduino takes in the data and outputs that random number + 1.</p>

Weekly task objective:

Q2:

```
6 baud_rate = 9600
7
8 # set serial port as suits your operating system
9 s = serial.Serial('COM4', baud_rate, timeout=10)
10
11 while True: # infinite loop, keep running
12
13     # a random number between 1 and 6.
14     data_send = random.randint(1, 6)
15
16     # Write in serial port to send to arduino
17     d1 = s.write(bytes(str(data_send), 'utf-8'))
18     print(f"Send >>> {data_send} ({d1} bytes)")
19
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
Recv <<< 4
Sleep for 4s
Send >>> 3 (1 bytes)
Recv <<< 1
Sleep for 1s
Send >>> 3 (1 bytes)
Recv <<< 4
Sleep for 4s
Send >>> 6 (1 bytes)
Recv <<< 4
Sleep for 4s
Send >>> 5 (1 bytes)
Recv <<< 1
Sleep for 1s
Send >>> 2 (1 bytes)
Recv <<< 2
Sleep for 2s
Send >>> 2 (1 bytes)
Recv <<< 5
Sleep for 5s
Send >>> 2 (1 bytes)
Recv <<< 3
Sleep for 3s
Send >>> 3 (1 bytes)
Recv <<< 3
Sleep for 3s
Send >>> 3 (1 bytes)
Recv <<< 3
Sleep for 3s
Send >>> 2 (1 bytes)
Recv <<< 3
Sleep for 3s
Send >>> 4 (1 bytes)
Recv <<< 4
Sleep for 4s
Send >>> 1 (1 bytes)
```

The log lines from the output shows the following: Firstly, Python sends a number from 1 to 6 to Arduino. Arduino takes in the number and blinks that number of times at 1 second interval. Then Arduino sends a random number between 1 to 6 to Python. Python then takes in the random number that is sent from Arduino and sleeps for that amount of time. The is the basic idea behind this task.

Q3:

Python:

```
import serial
import random
import time

# set baud rate, same speed as set in your Arduino sketch.
boud_rate = 9600

# set serial port as suits your operating system
s = serial.Serial('COM4', boud_rate, timeout=10)

while True: # infinite loop, keep running

    # a random number between 1 and 6.
    data_send = random.randint(1, 6)

    # Write in serial port to send to arduino
    d1 = s.write(bytes(str(data_send), 'utf-8'))
    print(f"Send >>> {data_send} ({d1} bytes)")

    # Read from serial port that arduino sends
    for _ in range(10): # retry 10 times
        d2 = s.readline().decode("utf-8").strip()
        if d2:
            print(f"Recv <<< {d2}")
            try:
                # Sleep based on the number arduino sends
                sleep_time = int(d2)
                print(f"Sleep for {sleep_time}s")
                time.sleep(sleep_time)
                break # exit the loop
            except ValueError:
                print(d2)
```

Arduino:

```
int x;

void setup() {
    Serial.begin(9600); // set baud rate
    pinMode(LED_BUILTIN, OUTPUT); // set up built in led as output
}

void loop() {
    while (!Serial.available()) {} // wait for data to arrive
```

```

    // read string data from Python and blink for that number of times at 1 second
interval
    x = Serial.readString().toInt();
    for(int i = 0; i < x; i++){
        digitalWrite(LED_BUILTIN, HIGH);
        delay(1000);
        digitalWrite(LED_BUILTIN, LOW);
        delay(1000);
    }

// Generate random number between 1 and 6
int ArduinoRandom = random(1,6);
Serial.println(ArduinoRandom);

// small delay to ensure data is sent properly
delay(50);

// Push the data through serial channel.
Serial.flush();
}

```

“From the Python code, Python sends a number from 1 to 6 to Arduino” corresponding to:

```

# a random number between 1 and 6.
data_send = random.randint(1, 6)

# Write in serial port to send to arduino
d1 = s.write(bytes(str(data_send), 'utf-8'))
print(f"Send >>> {data_send} ({d1} bytes)")

```

“Arduino takes in the number and blinks that number of times at 1 second interval” corresponding to:

```

    // read string data from Python and blink for that number of times at 1 second
interval
    x = Serial.readString().toInt();
    for(int i = 0; i < x; i++){
        digitalWrite(LED_BUILTIN, HIGH);
        delay(1000);
        digitalWrite(LED_BUILTIN, LOW);
        delay(1000);
    }

```

“Then Arduino sends a random number between 1 to 6 to Python” corresponds to:

```
// Generate random number between 1 and 6
int ArduinoRandom = random(1,6);
Serial.println(ArduinoRandom);

// small delay to ensure data is sent properly
delay(50);

// Push the data through serial channel.
Serial.flush();
```

“Python then takes in the random number that is sent from Arduino and sleeps for that amount of time” corresponding to:

```
# Read from serial port that arduino sends
for _ in range(10): # retry 10 times
    d2 = s.readline().decode("utf-8").strip()
    if d2:
        print(f"Recv <<< {d2}")
        try:
            # Sleep based on the number arduino sends
            sleep_time = int(d2)
            print(f"Sleep for {sleep_time}s")
            time.sleep(sleep_time)
            break # exit the loop
        except ValueError:
            print(d2)
```

Note that the for loop is for retrying to read the number sends from Arduino in case of empty string.

Q4:

Video: <https://www.youtube.com/watch?v=M5-kqivOLO4>

Q5:

Github repo is current set to private

tomadonna1 / SIT225_2024T2

CodeIssuesPull requestsActionsProjectsWikiSecurityInsightsSettings

SIT225_2024T2Private

Unwatch1Fork0Star0

main1 Branch0 Tags

Go to file

Add fileCode

tomadonna1 commit b4b4778 · now 3 Commits

Pass Task Hello Arduino commit now

.gitattributes Initial commit yesterday

README

Add a README

Add a README with an overview of your project.

Add a README

About

No description, website, or topics provided.

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Languages

C++ 53.6%Python 46.4%

Suggested workflows

Based on your tech stack

DjangoBuild and Test a Django ProjectConfigure

SLSA GenericConfigure

11°CPartly sunny

Search

ENGUS12:42 PM2-6/27/2024