

Student name: Yasmin Pokia

Student ID: s222245206

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
Pet/Cat feeder	Feeding times, amount of food dispensed, which food to dispense for the type of cat	Weight sensors to measure the amount of food, infrared sensors to detect the presence of the cat, motor controllers to operate the food dispenser mechanism
TxtBomber	Text input from the user	Buttons or a small keyboard for text input, position sensors to ensure accurate text printing, motor controllers to move the pens and print the text.
Sign language glove	Hand and finger movements	Flex sensors to measure the bending of fingers, accelerometers

		and gyroscopes to track hand orientation and movement, pressure sensors to detect touch and grip strength
--	--	-----------------------------------------------------------------------------------------------------------

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?

There is no sensing requirement in this system as this system does not require any external data collection through sensor data as it is built to function on a predetermined timing mechanism.

- b) What is the ‘thinking’ requirement in this system, if any?

The thinking requirement in this system is the timing logic on the Arduino microcontroller. To decide when to turn the LED on and off, the microcontroller must monitor the amount of time that has passed.

- c) What is the ‘acting’ requirement in this system, if any?

The acting requirement in this system is to control the state of the LED light, where the Arduino must control the LED by turning it on and off based on the timing logic.

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

- a) In Arduino-speak, what is a “sketch”?

In Arduino-speak, a “sketch” is a program created with the Arduino Software (IDE).

- 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 once at the very beginning of your program and never again, unless you reset or upload new code.

- ii) Which of the above two, is used to continuously run code over and over again?

The `loop()` function is used to continuously run code over and over again. The `loop()` function is called repeatedly once the `setup()` function is finished, enabling the Arduino to carry out its primary functions continually.

- c) What does **`pinMode()`** do?

Hint: <http://arduino.cc/en/Reference/HomePage>

The `pinMode()` function in Arduino is used to configure a specific pin to behave either as an input or an output. It sets the mode of the specified pin, defining whether it will be used to receive data (input) or send data (output).

d) What is a comment?

A comment in programming is a line or block of text in the code that is not executed by the program. Comments can be used to mark functionality for future reference, clarify the code, or offer explanations. They aid in the comprehension of the logic and purpose of the code by programmers. Comments are indicated with // for single-line comments and /*... */ for multi-line comments in Arduino sketches and many other programming languages.

e) What does the following line of code do:

```
delay(x);
```

Hint: <http://arduino.cc/en/Reference/HomePage>

The line 'delay(x)', pauses the execution of the Arduino program for 'x' milliseconds. The program doesn't go on to the following instruction during this period.

f) There is something you need to check before uploading your sketch. What is this?

Before uploading your sketch to the Arduino board, something you need to check is to make sure the correct Arduino board is selected in the Arduino IDE under 'Tools > Board', and the correct port is selected under 'Tools > Port' to communicate with your Arduino board.

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

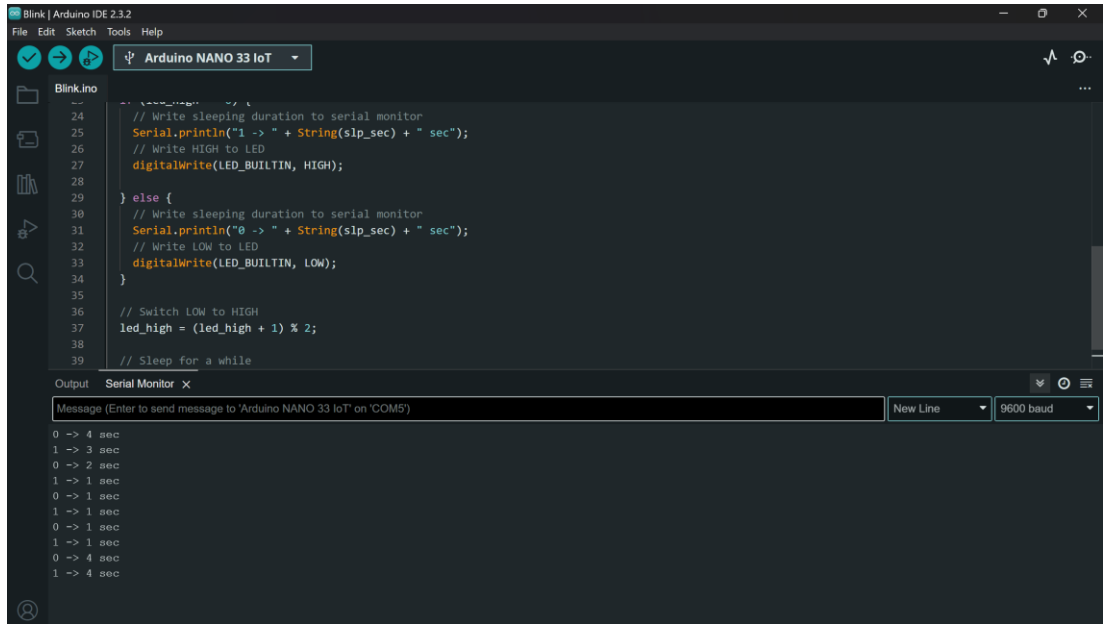
Steps	Actions
1	Identify the port through which Arduino is connected to your computer. You can find it in Arduino IDE Tools menu. Write Arduino sketch (or download from https://github.com/deakin-deep-dreamer/sit225/blob/main/week_1/sketch_blink.ino), which look like below, in Arduino IDE, deploy in your 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> 
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: The Serial.begin(6900) function initializes the serial communication at a baud rate of 6900 bits per second, which sets up the communication speed between the Arduino and the computer.</p> <p>The Serial.println(“message”) function sends a string message to the serial monitor, followed by a new line, which is used to print the current state of the LED (on or off) and the duration for which it will sleep.</p> <p>When the LED is turned on, the serial monitor prints ‘1 -> X sec’, where ‘X’ is the random sleep duration, whereas when the LED is turned off, the serial monitor prints ‘0 -> X sec’.</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: 8 bits in a byte, so 12 bytes is 96 bits. Time = number of bits / baud rate so 96 bits / 4800 bits per second = 0.02 seconds. Thus, it takes 0.02 seconds to send 12 bytes of data at a baud rate of 4800 bits per second.</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: No, there is no output in the serial monitor as the Arduino is waiting for data to arrive via the serial interface. The code has a 'while (!Serial.available()) {}' loop, which stops the execution until it receives data from the serial input.

To see any input in the serial monitor, type an integer in the input field and send it, and you will see a +1 increase in the integer you typed.

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

```

1  import serial
2  import random
3
4  # set baud rate, same speed as set in your Arduino sketch.
5  boud_rate = 9600
6
7  # set serial port as suits your operating system
8  s = serial.Serial('/dev/ttyACM1', boud_rate, timeout=5)
9
10 while True: # infinite loop, keep running
11
12     # a random number between 5 and 50.
13     data_send = random.randint(5, 50)
14
15     # write to serial port, set data encoding.
16     # Raw bytes are sent through serial ports, Python bytes() needs
17     # to know the encoding to generate bytes from string.
18     #
19     # We send a single integer which is read from Arduino sketch.
20     #
21     d = s.write(bytes(str(data_send), 'utf-8'))
22     print(f"Send >>> {data_send} ({d} bytes)")
23
24     # Read from serial port.
25     #
26     # readline keeps reading until a newline found in the data stream.
27     # Unlike write above, we just send an integer with no newline.
28     # You should receive data the same way as it is sent.
29     #
30     d = s.readline().decode("utf-8")
31     print(f"Recv <<< {d}")
32

```

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: When I run the Python script in command line, I receive an error.

	<p>Answer: The communication protocol used between the Arduino sketch and Python script is based on serial communication. The python script sends a random integer to the Arduino board, Arduino reads this integer, increments it by 1, and sends the result back to the Python script. Python script reads and prints the response.</p>
--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

SIT225 Data Capture Technologies

Pass Task: Hello Arduino!

Yasmin Pokia

S222245206

Task Objective

In this week, you have learned to write blink LED sketch for Arduino to periodically turn it ON/OFF and establish communication between Arduino and your computer through serial communication port. In this task, you are required to develop a communication protocol between Arduino and Python script running on your computer where Python script sends command to Arduino to blink LED a number of times and wait for Arduino to do the job and in response, Arduino sends a number back for which the Python script should wait that number of seconds before sending the next blink command and this process repeats.

Steps:

- 1. Python script runs and sends a random number through serial communication port where Arduino sketch is listening and receives the number. Python script should log the sending event in console with timestamp.*
- 2. Arduino blinks the LED that number of times with 1 second interval.*
- 3. Arduino writes to serial communication port a random number where the python script is listening and receives the number.*
- 4. Python script receives the number, logs in console with timestamp and sleeps that number of seconds. Logs in console when sleeping is done.*
- 5. Python script sends a number through the serial communication port same as step 1 and the process repeats forever.*

Q2. Perform the task mentioned above in the Task Objective and keep running for few minutes. Capture the screen showing Python console output of communication log as evidence of interaction with Arduino sketch. Describe the log lines in the screenshot in a paragraph following the image.

```
Administrator: C:\Windows\System32\cmd.exe
C:\Users\Yasmin Pokia\Downloads>serial_comm_script.py
2024-07-17 15:08:01 - Send >>> 2 (2 bytes)
2024-07-17 15:08:05 - Received <<< 8
2024-07-17 15:08:05 - Sleeping for 8 seconds
2024-07-17 15:08:13 - Sleeping done
2024-07-17 15:08:13 - Arduino finished its task.
2024-07-17 15:08:13 - Send >>> 2 (2 bytes)
2024-07-17 15:08:17 - Received <<< 8
2024-07-17 15:08:17 - Sleeping for 8 seconds
2024-07-17 15:08:25 - Sleeping done
2024-07-17 15:08:25 - Arduino finished its task.
2024-07-17 15:08:25 - Send >>> 2 (2 bytes)
2024-07-17 15:08:29 - Received <<< 7
2024-07-17 15:08:29 - Sleeping for 7 seconds
2024-07-17 15:08:36 - Sleeping done
2024-07-17 15:08:36 - Arduino finished its task.
2024-07-17 15:08:36 - Send >>> 1 (2 bytes)
2024-07-17 15:08:38 - Received <<< 9
2024-07-17 15:08:38 - Sleeping for 9 seconds
2024-07-17 15:08:47 - Sleeping done
2024-07-17 15:08:47 - Arduino finished its task.
2024-07-17 15:08:47 - Send >>> 3 (2 bytes)
2024-07-17 15:08:53 - Received <<< 5
2024-07-17 15:08:53 - Sleeping for 5 seconds
2024-07-17 15:08:58 - Sleeping done
2024-07-17 15:08:58 - Arduino finished its task.
2024-07-17 15:08:58 - Send >>> 3 (2 bytes)
2024-07-17 15:09:04 - Received <<< 8
2024-07-17 15:09:04 - Sleeping for 8 seconds
2024-07-17 15:09:12 - Sleeping done
2024-07-17 15:09:12 - Arduino finished its task.
2024-07-17 15:09:12 - Send >>> 3 (2 bytes)
2024-07-17 15:09:18 - Received <<< 7
2024-07-17 15:09:18 - Sleeping for 7 seconds
2024-07-17 15:09:25 - Sleeping done
2024-07-17 15:09:25 - Arduino finished its task.
2024-07-17 15:09:25 - Send >>> 5 (2 bytes)
2024-07-17 15:09:35 - Received <<< 6
2024-07-17 15:09:35 - Sleeping for 6 seconds
2024-07-17 15:09:41 - Sleeping done
2024-07-17 15:09:41 - Arduino finished its task.
2024-07-17 15:09:41 - Send >>> 1 (2 bytes)
2024-07-17 15:09:43 - Received <<< 6
2024-07-17 15:09:43 - Sleeping for 6 seconds
2024-07-17 15:09:49 - Sleeping done
2024-07-17 15:09:49 - Arduino finished its task.
2024-07-17 15:09:49 - Send >>> 5 (2 bytes)
2024-07-17 15:09:59 - Received <<< 9
2024-07-17 15:09:59 - Sleeping for 9 seconds
2024-07-17 15:10:08 - Sleeping done
2024-07-17 15:10:08 - Arduino finished its task.
2024-07-17 15:10:08 - Send >>> 2 (2 bytes)
```

The communication between the Arduino and the Python script is recorded in the log. Every log entry begins with the action completed and is followed by a timestamp. The Arduino receives a random integer (Send >>> x (2 bytes)) from the Python script, where x is the provided integer. After processing this integer, the Arduino waits x seconds, blinks the LED x times, and then returns a random integer (Received <<< y), where y is the integer that the Python script received. After logging the receiving integer, the Python script pauses for x seconds before iterating through the loop once more. The log entry indicating that "Arduino finished its task." verifies that the Arduino has done flashing and is ready for the Python script to submit the next number. This constant communication keeps the Python script and the Arduino in sync, proving that the given task was successfully completed.

Q3. Paste Python and Arduino sketch and explain program steps. Your explanation should match the description you prepared for Q2.

Python Program Explanation:

```
import serial
import random
import time
from datetime import datetime

# Set baud rate, same speed as set in your Arduino sketch.
```

```

baud_rate = 9600

# Set serial port as suits your operating system
s = serial.Serial('COM5', baud_rate, timeout=5)

def log(message):
    print(f"{datetime.now().strftime('%Y-%m-%d %H:%M:%S')} - {message}")

while True: # Infinite loop, keep running
    # Generate a random integer between 1 and 5.
    data_send = random.randint(1, 5)

    # Write the generated integer to the serial port, encode it as UTF-8.
    d = s.write(bytes(str(data_send) + '\n', 'utf-8'))
    log(f"Send >>> {data_send} ({d} bytes)")

    # Read from the serial port until "DONE" is received.
    while True:
        d = s.readline().decode("utf-8").strip()
        if d == "DONE":
            log("Arduino finished its task.")
            break
        if d.isdigit():
            wait_time = int(d)
            log(f"Received <<< {wait_time}")

            # Sleep for the received amount of seconds.
            log(f"Sleeping for {wait_time} seconds")
            time.sleep(wait_time)
            log("Sleeping done")

```

1. **Serial Initialisation and Logging:** A serial connection (s) to COM5 are initialised by the Python script at a baud rate of 9600. It specifies the log function, which prints timestamps along with messages.
2. **Main Loop Execution:** To maintain constant communication with the Arduino, the script enters an endless loop (while True):
 - **Create Random number:** It creates a data_send random number, ranging from 1 to 5.
 - **Data Send to Arduino:** This action is logged and a random integer is sent to the Arduino via the serial port.
 - **Waiting for Arduino Response:** It goes into another loop to read data from the serial port until it gets the "DONE" signal from the Arduino, which means that the task has been completed. When a number is input, the Arduino understands it as the amount of time (in seconds) that it will blink its LED and go to sleep.
 - **Handle Arduino Response:** The script logs a numeric value as "Received" and then goes to sleep for the specified number of seconds after getting it.
3. **Logging:** The script records all of the actions that are done during the process, such as transmitting data, getting acknowledgements, and sleeping.

Arduino Sketch Explanation:

```
int ledPin = 13; // Set the LED pin (e.g., built-in LED on pin 13)
int x;

void setup() {
  Serial.begin(9600); // Set baud rate
  pinMode(ledPin, OUTPUT); // Set the LED pin as output
}

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

  // Read string data from Serial and ignore newline/carriage return
  characters
  String input = Serial.readStringUntil('\n');
  x = input.toInt();

  // Debugging print to ensure the correct value is received
  Serial.print("Received: ");
  Serial.println(x);

  // Wait for x seconds before blinking the LED
  for (int i = 0; i < x; i++) {
    delay(1000); // Wait for 1 second each loop iteration
  }

  // Blink the LED 'x' times
  for (int i = 0; i < x; i++) {
    digitalWrite(ledPin, HIGH); // Turn the LED on
    delay(500); // Wait for half a second
    digitalWrite(ledPin, LOW); // Turn the LED off
    delay(500); // Wait for half a second
  }

  // Send a random number back to Python
  int response = random(5, 10);
  Serial.println(response);

  // Indicate that the Arduino is ready for the next command
  Serial.println("DONE");
  Serial.flush();
}
```

1. Initialisation: To initialise, the Arduino configures pin 13 as an LED output and establishes serial communication at a rate of 9600 baud.
2. Main Loop Execution: The loop() method on the Arduino runs continuously:
 - Waiting for Serial Data: This process waits for the serial port to receive data (Serial.available()).
 - Read and Parse Data: Until it comes across a newline (\n), it reads the incoming string from the serial port. It uses toInt() to convert this string to an integer (x).
 - Debugging Print: To aid with debugging, it prints the integer that was received.
 - Delay Before Blinking: The LED doesn't start to blink for x seconds.
 - Blink the LED: It turns the LED on and off with half-second delays (delay(500)) by blinking the LED x times.
 - Send Response to Python: Following its blinking, it creates a random integer (response) between 5 and 10 and serially communicates it back to the Python script.
 - Signal Completion: It sends "DONE" to the Python script to let it know that the blinking is complete and that it is prepared to receive another command.
 - Make sure all data is transmitted out of the serial buffer by using the flush serial buffer function (Serial.flush()).

Q4. Create a video in Panopto/Cloud Deakin showing your program execution and any instruction to be followed in order to run your code, share the link in your report.

<https://deakin.au.panopto.com/Panopto/Pages/Viewer.aspx?id=a2b88143-e96c-4e33-a7fd-b1b00080a122>

Q5. Create a directory 'SIT225_2024T2' in your drive and a subdirectory 'week-1' where you copy the Python script file and Arduino sketch file. Create a new private repository on GitHub sharing 'SIT225_2024T2' folder. You will be creating new subfolders every week. Include the link to your repository in your report with a GitHub page screenshot of weekly folder content. A tutor may try to access your GitHub link, if necessary. Give access to your tutor by adding tutor's email address as a collaborator of your private repository.

https://github.com/YP682/SIT225_2024T2

github.com/YP682/SIT225_2024T2

Gmail

DeakinSync

Homepage - Cloud...

OnTrack

chatgpt

Paraphrasing Tool ~...

edclub

JustinGuitar Dashbo...

All Bookmarks

YP682 / SIT225_2024T2

Type to search

+ 🔍 📄 📁 🏠

<> Code

Issues

Pull requests

Actions

Projects

Security

Insights

Settings

SIT225_2024T2

Private

Unwatch 1

Fork 0

Star 0

main 1 Branch 0 Tags

Go to file

Add file

Code

YP682

Initial commit: Added Python script and Arduino sketch for Week 1

7490dc · 6 minutes ago

1 Commit

1.1P.docx

Initial commit: Added Python script and Arduino sketch for ...

6 minutes ago

1.1P.ino

Initial commit: Added Python script and Arduino sketch for ...

6 minutes ago

1.1P.py

Initial commit: Added Python script and Arduino sketch for ...

6 minutes ago

Activity_week_1.docx

Initial commit: Added Python script and Arduino sketch for ...

6 minutes ago

Activity_week_1.pdf

Initial commit: Added Python script and Arduino sketch for ...

6 minutes ago

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++ 33.9%

Python 46.1%

Suggested workflows

Based on your tech stack

Django

Build and Test a Django Project

Configure

Python package

Create and test a Python package on multiple Python versions.

Configure

Publish Python Package

Publish a Python Package to PyPI on release.

Configure

More workflows

Dismiss suggestions

© 2024 GitHub, Inc.

Terms

Privacy

Security

Status

Docs

Contact

Manage cookies

Do not share my personal information