Lab Mini-Project:

Arduino Gadget Companion

Submitted To:

Lori Hogan

ENGI 1020 – Introduction to Programming

Memorial University of Newfoundland

Submitted By:

Shawn Adrian

sadrian@mun.ca - 201830718

Date Submitted:

August 11, 2020

Table of Contents

Introduction

This report details an Arduino-based project written in Python 3 that utilizes the Grove Starter Kit for Arduino. The “Arduino Gadget Companion” project is a simple gadget that actively displays the time, temperature, and has an interactive feature called the “Mind Reader”, all of which are triggered by user input. The purpose of the project is to utilise the Nanpy firmware and the engi1020 module introduced in the first lab session to apply key concepts learned throughout the course.

The goal of the project is to have a companion device, similar to a digital clock, that also has a customisable interactive feature, which by default in this case is the “Mind Reader”. Like a locked mobile phone, the gadget will display an “inactive” screen, prompting the user to register input before displaying an active clock. After the gadget enters the “active” screen, the user will have the choice between two input devices to display either the current temperature or the interactive feature.

Final Design Outline

Comprehensively described using mathematical expressions, flow charts, pseudocode, and other appropriate methods.

The key components of the script revolve around the important roles of the Arduino input devices. When the script is run, the user sees cycling colours in the LCD and is prompted to press the button to start displaying time. The time display is defined as a function that refreshes every half a second as a while loop waiting for either one of the input devices to be triggered. The temperature and “mindReader” functions are each defined in a function that would display a related string for a few seconds before returning to the time display. Figure 1 illustrates the overall algorithm of the script as a flow chart drawn using DRAW.io.

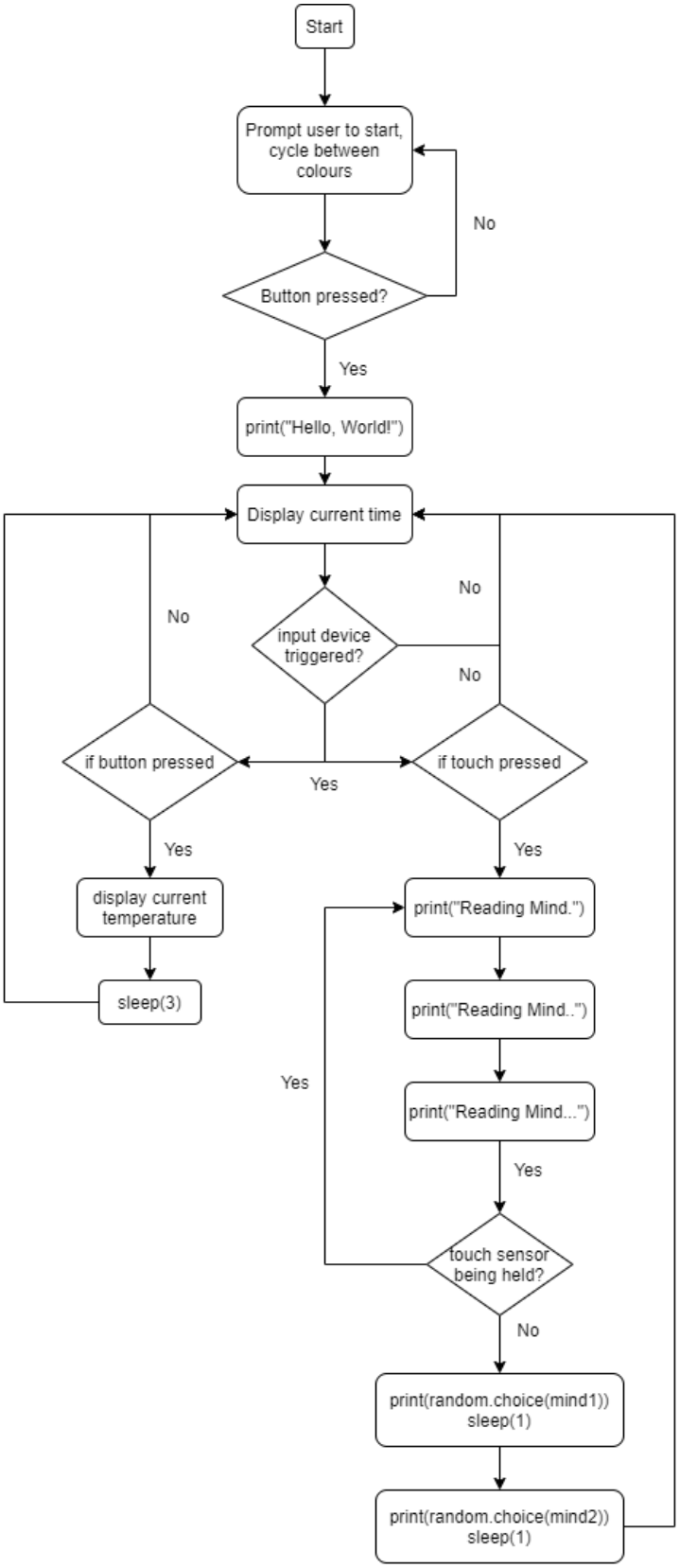

Figure 1: Flow chart describing the process of the script

Function Specifications

The overall script heavily utilises functions, both defined and from imported modules. The functions listed below are from imported modules and functions that are built-in python.

- ***engi1020.arduino module:***
    - ***lcd_hsv(hue, saturation, brightness):*** Used to control hue, saturation, and brightness of the LCD output.
    - ***lcd_print("string"):*** Used to print a string onto the LCD. The LCD however only outputs 16 characters, so any string longer than 16 characters would be cut at the 16th character.
    - ***lcd_clear():*** Used to clear any string printed onto the LCD.
    - ***digital_read(pin slot):*** Used to read the input from a digital device
    - ***temp_celsius():*** Used to output the reading given by the temperature sensor in Celsius
- ***time module:***
    - ***sleep(time):*** Used to pause the script for a given time interval
    - ***asctime():*** Used to print the current given day, date, time, and year
- ***random module:***
    - ***choice(list):*** Used to select a random element of a given list
- ***round(x, d.p.):*** Used to round a float value to a given number of decimal places

Images Printed

The images presented below are the desired outputs printed on the Arduino LCD

Figure 2: Scrolling text and alternating colours

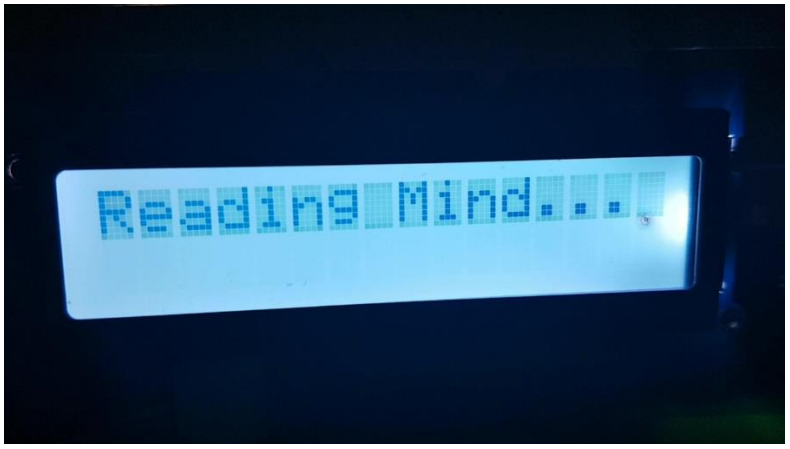

Figure 3: String printed when touch sensor is held

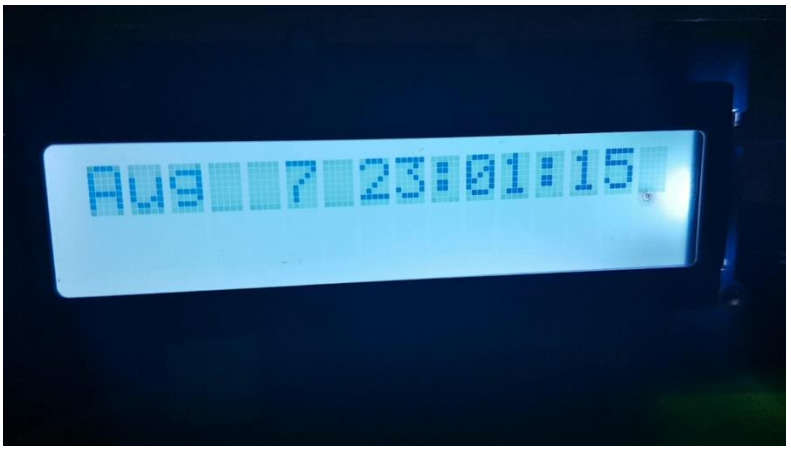

Figure 4: Time and Date display

Test Plans

Several tests were run to determine if the script would properly output the expected values into the Arduino setup.

Table 1: Controlled test for "inactive" screen

Controlled	Press button to exit the start screen and display time (active screen)	Let the colour loop continue without interference	Test and Implement scrolling text
Expected Output	Either the input immediately triggers a loop break, or the input doesn't come fast enough for it to register	The LCD would continue to loop between the colours with no issue	A given script would scroll through the LCD either every few seconds or every colour change
Actual Output	Initially, an if statement was put near the end of the loop, causing nothing to happen until the script passes through all colours to decide whether to loop again or not.	Output as expected	LCD doesn't scroll text, text overlaps

Changes Made	Added an if statement after each prompt to change colour to break out of the loop:
	`lcd_hsv(0,1,255)`
	`lcd_clear()`
	`lcd_print("Hold button to s")`
	`sleep(1)`
	`if digital_read(d1) == True:`
	`  break`
	Also changed the prompt from "Press button" to "Hold button" to let sleep function to finish running before the input is read to decide to start or not
		Started with 3 colours but went with 6 colours to attempt to create a scroll effect with the printed string "Hold button to start.". This is because the LCD would only fit 16 characters, and the printed string is 21 characters long	Since the LCD only fits 16 characters, lcd_clear was used to clear out the screen and a string "cut" to only 16 characters is alternated between colours to give a scrolling effect

Table 2: Controlled test for "active" screen and features

Controlled	Press either one of the inputs to initiate a feature	Press the button to display the current temperature and manipulate the temp sensor	Press the touch sensor to display the "mind reader" function
Expected Output	The script would run either the mindReader or tempNow function	Display temp when triggered, back to time, then display a different temp	The function would activate and not give an output as long as the button is held (i.e. if bInput == True)
Actual Output	The script doesn't register any input. Changes were made, and the script would stop after a either one of the functions ran.	Temperature would display but wouldn't change, likely because the temp input function is outside the display function as a global variable	The function wouldn print "reading mind" only for a few seconds even if the button is let go (i.e. triggers as long as there is input). If the button is held long enough, the script would stop running after the result is printed

Changes Made	The function checkInput was made to determine which input device was pressed. It would return the slot of the input.	A new variable was created within the temperature function to update only when the button is pressed and round the temperature float input to 2 decimal places
		`temp = temp_celsius(a1)`
		`rTemp = round(temp, 2)`
		
		`lcd_clear()`
		`lcd_print("It's ")`
		`lcd_print(rTemp)`
		`lcd_print(" C")`
		`sleep(3)`
			Added a while loop to not give a result until the user lets go of the touch sensor, which also prevented the script from completely stopping. This is more in line with the initial plan
			`while digital_read(d2) == True:`
			`  lcd_clear()`
			`  lcd_print("Reading Mind.")`
			`  sleep(0.5)`
			`  lcd_clear()`
			`  lcd_print("Reading Mind..")`
			`  sleep(0.5)`
			`  lcd_clear()`
			`  lcd_print("Reading Mind...")`
			`  sleep(0.5)`
			
			`  if digital_read(d2) == False:`
			`  lcd_clear()`
			`  lcd_print(random.choice(mind1))`
			`  sleep(1)`
			
			`  lcd_clear()`
			`  lcd_print(random.choice(mind2))`
			`  sleep(1)`
			`  break`

Link to key concepts

- **Expressions**
    - The function *temp_celsius* from the *engi1020.arduino* module utilises mathematical expressions that converts input from the temperature sensor from resistance to Celsius. The function returns the temperature as a float.
- **Variables**
    - Local variables are used within the *tempNow* function for the time operation to calculate and round off temperature and in the *timeNow* function.
- **Flow Control**
    - If statements are used in the *startScreen* function to break out of the while loop given the condition that the button is pressed.
    - An if statements is used in the *mindReader* function to break out of a while loop that prints a string to then print a random string from a list.
    - If statements are used in the *checkInput* function to return a value under the condition that the button or touch sensor is pressed.
    - If statements are used in the *timeDisplay* function to either run the *tempNow* function or the *mindReader* function depending on which input device is triggered.
- **Loops**
    - Use of a while loop to cycle between colours and string printed into the LCD in the startScreen function. The loop is broken if the button is pressed.
    - A while loop is used in the *mindReader* function to continuously print “Reading Mind…” while the touch sensor is held.

- A while loop is used to continuously refresh the time displayed on the LCD in the timeDisplay function.

- **Strings**

  - In the *startScreen* function, a string of words is repeated and alternated through each colour to give a scrolling effect.
  - A selection of strings is put into a list and are then printed onto the LCD.
  - Most of the outputs are strings

- **Lists and Arrays**

  - Separate lists are used to store a list of words that randomly chosen and then printed in the *mindReader* function.

- **Functions and Modules**

  - Functions from the *engi1020.arduino* module plays an important role in receiving input and delivering output to the Arduino device.
  - The *sleep* function from the *time* module is used to allow the LCD to display certain strings or outputs for a few seconds before it changes to a different ouput.
  - The *choice* function from the *random* module is used to print a random string from a list of string in the *mindReader* function.
  - A total of 6 functions were defined for the script, all of which feature the functions present in the script.

Challenges, Triumphs and Changes

One of the most challenging part of the lab project is figuring out how to configure the temperature sensor and its function. By default, the temperature sensor inputs an analog integer value of resistance. The initial plan was to create mathematical expressions that would convert resistance to Celsius based of the following formulae:

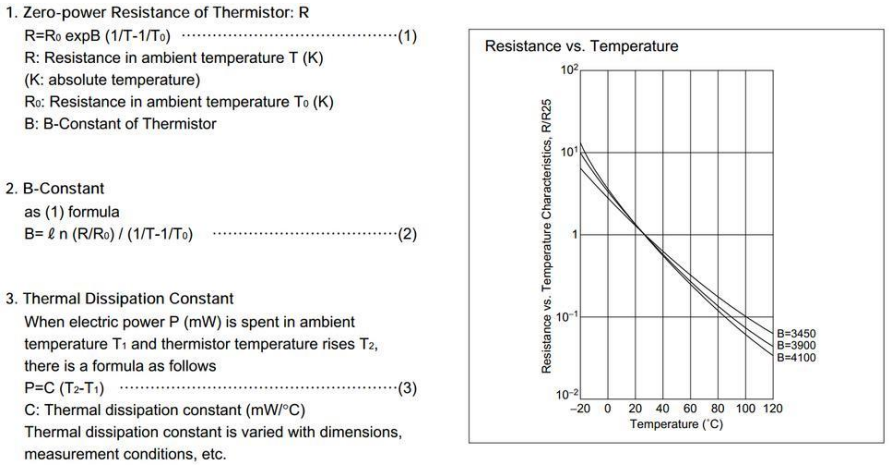

Figure 5: Constants, formulae, and a graph of the relation between resistance and temperature

The initial issue was implementing the maths to Python code, which required information on the temperature sensor, which was gathered from the seedstudio wiki. The grove temperature used is v1.2, with the specifications:

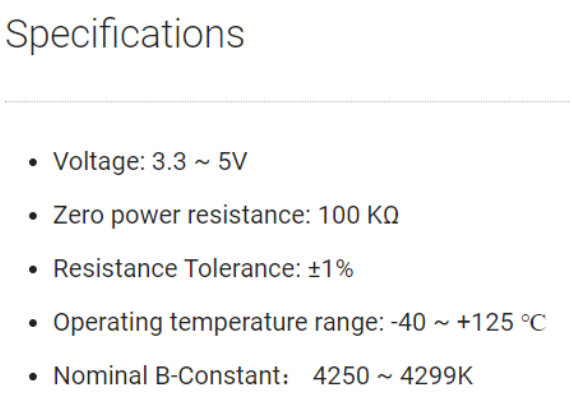

Figure 6: Specifications of the temp sensor v1.2

To make things easier, there was an existing Arduino code written in C++. It wasn't much of an issue modifying the script to be written in python, the actual issue was getting the script to output the correct value. Say typical room temperature is around 25 C, the temperature sensor would instead print a value closer to temperature in Kelvin, but negative instead of positive.

```
// Demo code for Grove - Temperature Sensor V1.1/1.2
// Loovee @ 2015-8-26

#include <math.h>

const int B = 4275;               // B value of the thermistor
const int R0 = 100000;            // R0 = 100k
const int pinTempSensor = A0;     // Grove - Temperature Sensor connect to A0

#if defined(ARDUINO_ARCH_AVR)
#define debug  Serial
#elif defined(ARDUINO_ARCH_SAMD) ||  defined(ARDUINO_ARCH_SAM)
#define debug  SerialUSB
#else
#define debug  Serial
#endif

void setup()
{
    Serial.begin(9600);
}

void loop()
{
    int a = analogRead(pinTempSensor);

    float R = 1023.0/a-1.0;
    R = R0*R;

    float temperature = 1.0/(log(R/R0)/B+1/298.15)-273.15; // convert to temperature

    Serial.print("temperature = ");
    Serial.println(temperature);

    delay(100);
}
```

Figure 7: C++ Script to convert resistance to Celsius

After several failed attempts at getting the temperature script to work properly, the next best decision is to consult help. Thankfully, the professor was able to assist in the matter and mentioned that there was an already existing function that does the mathematical calculations; temp_celsius(). This not only made the process in creating the temperature function much easier, it also made the script much shorter by utilising the engi1020.arduino module.

If there exists an opportunity to build upon the project, getting the gadget to go back to the start screen when idling for a set amount of time would be top priority. Besides that, if given access to more input devices, a function to stop the script as if "turning off" the gadget would also be interesting.

The greatest accomplishment is getting the mindReader function to continuously loop and print the script "Reading mind…" as long as the button is held. This was the most satisfying success because it went through the most trial and error. Referring back to course notes, assignments, and labs ultimately helped in finding a solution for this particular problem.

References

[1] B. Zuo, "Grove - Temperature Sensor V1.2 - Seeed Wiki", Wiki.seeedstudio.com, 2020. [Online]. Available: https://wiki.seeedstudio.com/Grove-Temperature_Sensor_V1.2/.

[2] "Grove - Temperature Sensor V1.2", Seeeddoc.github.io, 2020. [Online]. Available: https://seeeddoc.github.io/Grove-Temperature_Sensor_V1.2/.