DEVOPS FOR AIOT
SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING, SINGAPORE POLYTECHNIC

LABORATORY 5: PYTHON PROGRAMMING FOR AIOT DEVELOPMENT

Objectives

By the end of the laboratory, students will be able to

- Introduction to the following peripherals on the Raspberry Pi Development board
    - Keypad
    - LED
    - LCD
    - Buzzer
    - Servo Motor
    - ADC (Analog to Digital Converter)

Activities

- Introduction to Raspberry Pi and evaluation board peripherals
- Implementation of Python code to interface with the keypad and LCD
- Interfacing with the ADC (Analog to Digital Converter)
- Implementation of Python code to control the Servo motor
- Simple application to use the Keypad, LCD, ADC, Buzzer and Servo Motor

Review

- Implemented Python code to interface with the various peripherals connected to the Raspberry Pi Development Board

Equipment:

- Windows OS laptop
- Raspberry Pi + Development Board
- USB-C 5 volt power adapter
- 9 volt power adapter
- USB to Ethernet adapter

1 Keypad and LCD

To interact with the user, the Raspberry Pi Development board uses the LCD display and a numeric keypad to display information and to get inputs from the user.

Figure 1

1.1 Clone the git repository from the URL https://github.com/ET0735-DevOps-AIoT/Lab5.git **including the HAL Git submodule**

1.2 Open the PyCharm project "Lab5" and configure the necessary settings to remotely debug and run remotely on the Raspberry Pi

1.3 Refer to the steps in Lab 4 if you need information on how to setup PyCharm/VsCode to remotely debug and run on the Raspberry Pi.

1.4 Open the Lab5 PyCharm project you just cloned from Github and run the Python file main.py which implements a basic Python application to scan the based on the example Github Wiki page below,

https://github.com/ET0735-DevOps-AIoT/raspberry-pi-dev-board/wiki/Keypad

1.5 Execute and run the Lab5 main.py Python file and observe that the key pressed is displayed in the PyCharm console shown in Figure 2 below

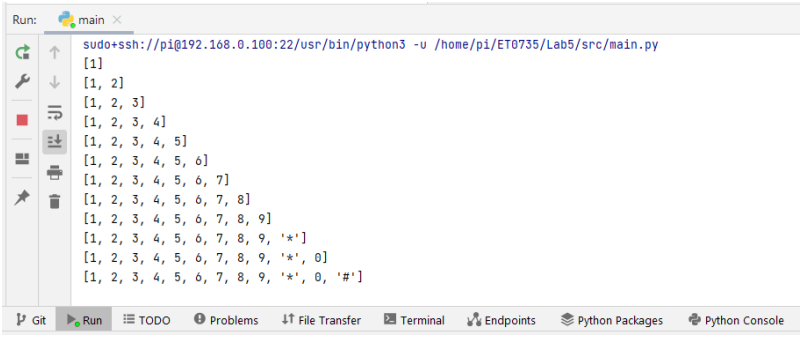

Figure 2

2 Python Threads

Until this point we have been developing Python code that runs in a single thread execution.

In Lab 2, we have developed Python applications to start from a single "main" entry point and executed all functions sequentially.

However, there are many scenarios where executing only from a single main thread may be inefficient and cause significant delays in the Python application.

2.1 In the file "led_control.py", add the following code from Table 1

```
def led_thread():
    global delay

    delay = 0

    while(True):
        if delay != 0:
            led.set_output(20,1)
            sleep(delay)
            led.set_output(20, 0)
            sleep(delay)
```

Table 1

2.2 Notice that in the function "led_thread()", there is an infinite loop that runs forever and blinks the LED at a fixed rate based on the variable "delay"

2.3 Also observe that the variable "delay" has the Python keyword "global" before it is declared

2.4 The "global" keyword in Python is used to define variables that have a global scope across a file and can therefore be read or modified anywhere in the same Python file

```
def led_control_init():

    global delay

    t1 = Thread(target=led_thread)

    t1.start()

    #Set initial LED blinking every 1 second after Thread starts
    delay = 1
```

Table 2

2.5 The code above is used to start a Python thread

2.6 In the file "LED_control.py", add the code in above to start the thread and write your observations in the box below

2.7 In the Python file "main.py", call the function "led_control_init()" defined in "led_control.py" and observe what happens to the LED on the Development Board.

Exercise 1

In this exercise, we will implement a simple Python application running on the Raspberry Pi that allows the user to control the blinking frequency of the LED using the LCD and Keypad.

1. The callback function "key_pressed()" is called whenever a key is pressed on the keypad.

   Note: Refer to https://pythonexamples.org/python-callback-function/ for more information on "callback functions"

2. Adapt the implementation in the function "key_pressed()" to control the blinking of the LED based on the requirements defined in Table 3 below,

Requirement ID	Requirement
REQ-01	When the application starts, the following test shall be displayed on the LCD screen Line 1 = "LED Control" Line 2 = "0:Off 1:Blink"
REQ-02	If the user presses the key 1, then the LED shall blink every 1 second and the following shall be displayed om the LCD Line 1 = "LED Control" Line 2 = "Blink LED"
REQ-03	If the user presses the key 0, then the LED shall be turned OFF and stop blinking Line 1 = "LED Control" Line 2 = "OFF LED"

Table 3

3 Analog to Digital Converter (ADC)

The ADC module allows us to interface with analog sensors and devices that we might need to monitor in a real world application.

3.1 The Raspberry Pi 4 board does not contain an on-board ADC module, therefore on the Development Board, an external IC is connected to the Raspberry Pi via an SPI (Serial Peripheral Interface.)

3.2 This allows the Raspberry Pi to interface with analog sensors and devices and we can read the ADC values in our Python code

3.3 On the Raspberry Pi Development Board, we have the following ADC channels listed in Table 4 below,

ADC Channel	Analog Device
0	LDR (Light Dependent Resistor)
1	Potentiometer

Table 4

3.4 Study the implementation of the file "hal_adc.py" and identify the 2 functions that initializes and reads the ADC value.

Function to initialize ADC = ADC.init() Function to read ADC channel = ADC.get_adc_value(adcnum)

3.5 Also identify which functions can be used in the file "hal_adc.py" to read the raw ADC values based on the "ADC Channel" number in Table 5

4 Servo Motor

4.1. Servo motors are used for applications which require angular control of the position of the motor.

4.2. The Raspberry Pi Development Board has servo motor which is controlled using PWM (Pulse Width Modulation).

4.3. By changing the Duty Cycle of the signal to the Servo motor, the Servo position can then be controlled.

4.4. Study the implementation of the file "hal_servo.py" and identify the functions that will initialize the servo motor Software Driver and the function that can be used to control the angular position of the Servo Motor.

Write your answers in the box below,

```
from hal import hal_servo as servo
servo.init()
servo.set_servo_position(position)
```

Exercise 2

In this exercise we will implement a Python application that controls the angle of the servo motor based on the analog voltage read from the potentiometer on the Raspberry Pi Development board

1. In the PyCharm project "Lab5", create a new file "adc_servo_controller.py"

2. Based on the graph in Figure 3 below, implement Python code to control the Servo Motor based on the ADC potentiometer on the Raspberry Pi Development Board.

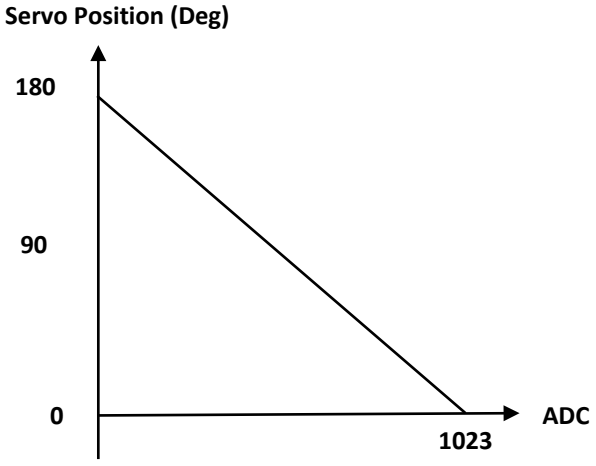

Figure 3

Exercise 3

In this exercise, we will implement a simple "Clock" that displays the current system time and date on the LCD.

1. The sample code below uses the Python "time" library to get the current system time from your laptop or Raspberry Pi OS.

```
import time

local_time = time.localtime() # get struct_time
time_string = time.strftime("%H:%M:%S", local_time)
```

Table 5

2. In the PyCharm "Lab5" project, create a new Python file "clock.py"

3. Implement a Clock application in python based on the requirements in Table 6 below

4. For the implementation, you may consider to use Python Threads to make the code run more efficiently

Requirement ID	Requirement
REQ-01	The LCD shall display the following strings below and the time displayed on the LCD be updated every 1 second LCD Line 1 = "hh : mm : ss" LCD Line 2 = "dd : mm : yyyy"
REQ-02	The colon characters ':' displayed for the time should be blinked every 1 second.

Table 6

Exercise 4

In this exercise, we will implement a simple "Electronic Safe" that allows the user to enter a 4 digit PIN

1. In the PyCharm "Lab5" project, create a new Python file "electronic_safe.py"

2. Implement a "Electronic Safe" application in python based on the requirements in the Table 7 below

Requirement ID	Requirement
REQ-01	The LCD shall display the following strings LCD Line 1 = "Safe Lock" LCD Line 2 = "Enter PIN: "
REQ-02	If the user presses any buttons on the keypad, the LCD shall display on the 2nd line the asterisk '*' character and not the actual digit pressed on the keypad.
REQ-03	If the user enters the correct PIN, then the LCD shall display the following LCD Line 1 = "Safe Unlocked" LCD Line 2 = ""
REQ-04	If the user enters the wrong PIN, then the LCD shall display the following and activate the buzzer for 1 second LCD Line 1 = "Wrong PIN" LCD Line 2 = ""
REQ-05	If the user enters the wrong PIN 3 times, then then the LCD shall display the following, LCD Line 1 = "Safe Disabled" LCD Line 2 = ""

Table 7