BANGABANDHU SHEIKH MUJIBUR RAHMAN AVIATION AND AEROSPACE UNIVERSITY (BSMRAAU)

Real Time Embedded System Sessional

Experiment No. 03: Understand the configuration of BLYNK Server and IoT based LED control using Real time embedded system design.

Objectives: The objectives of this experiment are-

a) Learn how to use Blynk Server
b) To understand the real time control system
c) To understand the IoT concept

Equipment:

a) Embedded System module ES-101
b) Desktop Computer/ Laptop
c) Smart Phone

Theory:

Designing a Blynk and Raspberry Pi LED control system involves integrating hardware and software components to allow remote control of LEDs through a mobile application.

Blynk Server is a pivotal component of the Blynk IoT platform, facilitating real-time communication between mobile applications and various IoT devices, such as Raspberry Pi, Arduino, and other microcontrollers. It operates on a client-server architecture, where the Blynk app acts as the client that sends commands and receives data from connected devices via the server.

The server supports both cloud and local deployment options, allowing users to choose between the convenience of a hosted solution and the privacy of a self-hosted server. Key features include device management, real-time data exchange, and a rich set of app widgets, enabling users to create interactive and responsive IoT applications. The Blynk Server also provides APIs for integration with external services, enhancing its versatility in various applications, from home automation and industrial monitoring to smart agriculture and prototyping.

Library Details

A. BlynkLib

BlynkLib with a Raspberry Pi, first need to install the library and set up your project to connect your Raspberry Pi to the Blynk platform. Here’s a step-by-step guide to get you started:

1. **Install the Required Packages**

```sh
pip3 install blynk-library
```

2. **Example code**

```python
import BlynkLib
import time

# Your Blynk authorization token
BLYNK_AUTH = 'YourAuthToken'

# Initialize Blynk
blynk = BlynkLib.Blynk(BLYNK_AUTH)

@blynk.on("connect")
def on_connect():
    print("Connected to Blynk!")

@blynk.on("virtual_read/V0")  # Example for reading a virtual pin
def read_virtual_pin_handler(value):
    print(f"Received value from V0: {value}")

while True:
    blynk.run()  # Maintain connection
    time.sleep(1)
```

B. RPi.GPIO

The `RPi.GPIO` library is a popular Python module used for interfacing with the GPIO (General Purpose Input/Output) pins on Raspberry Pi. It allows you to control hardware components like LEDs, buttons, sensors, and motors directly from your Raspberry Pi using Python scripts.

Key Features of RPi.GPIO:

1. **Pin Control**: Easily set pins as input or output and read or write digital signals.
2. **Event Detection**: Monitor pin states for changes (e.g., button presses) using callbacks.
3. **PWM Support**: Control the brightness of LEDs or speed of motors through Pulse Width Modulation.
4. **Flexible Pin Numbering**: You can use both physical pin numbers and BCM (Broadcom) GPIO numbers.

C. BLYNK AUTH TOKEN : The BLYNK_AUTH_TOKEN is a unique authentication token used in the Blynk IoT platform. Blynk is a popular platform for building Internet of Things (IoT) applications, allowing users to connect devices, sensors, and various hardware to the Blynk app for monitoring and control.

The BLYNK_AUTH_TOKEN is essential for:

1. **Device Authentication**: It ensures that only authorized devices can communicate with the Blynk server.
2. **App Communication**: It enables the Blynk app to send commands to the device and receive data from it.

Pin configuration

LED			
LED 1	**LED 2**	**LED 3**	**LED 4**
23	**24**	**25**	**1**

Circuit Diagram

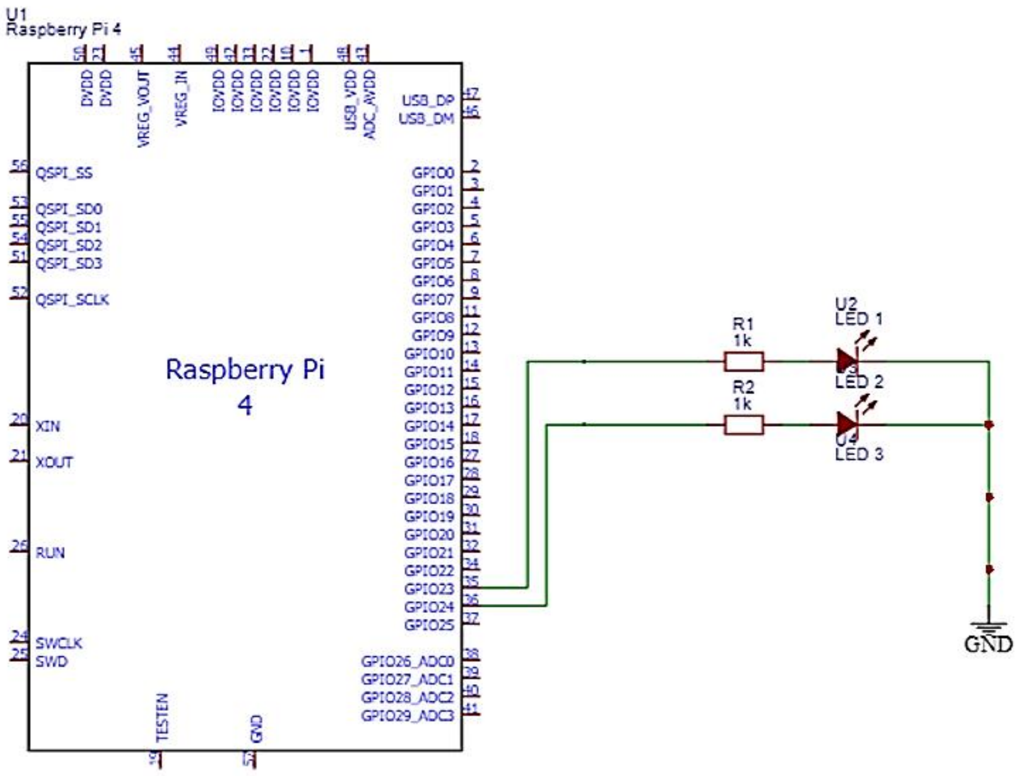


Figure 1: Circuit Diagram

Code:

```
import BlynkLib
import RPi.GPIO as GPIO
from BlynkTimer import BlynkTimer

BLYNK_AUTH_TOKEN = '00mWla41C4sT6SsT2b7Tr1gd5d7lvXek'
led1 = 23
led2 = 24
GPIO.setmode(GPIO.BCM)
GPIO.setup(led1, GPIO.OUT)
GPIO.setup(led2, GPIO.OUT)
x = 20
# Initialize Blynk
blynk = BlynkLib.Blynk(BLYNK_AUTH_TOKEN)

# Led control through V0 virtual pin
@blynk.on("V0")
def v0_write_handler(value):
#    global led_switch
    if int(value[0]) is not 0:
        GPIO.output(led1, GPIO.HIGH)
        print('LED1 HIGH')
    else:
        GPIO.output(led1, GPIO.LOW)
        print('LED1 LOW')
# Led control through V0 virtual pin
```

```
@blynk.on("V1")
def v1_write_handler(value):
#    global led_switch
    if int(value[0]) is not 0:
        GPIO.output(led2, GPIO.HIGH)
        print('LED2 HIGH')
    else:
        GPIO.output(led2, GPIO.LOW)
        print('LED2 LOW')
#function to sync the data from virtual pins
@blynk.on("connected")
def blynk_connected():
    print("Raspberry Pi Connected to New Blynk")
while True:
    blynk.run()
```

Code Explanation:

This code snippet is designed to control two LEDs connected to a Raspberry Pi using the Blynk IoT platform. Here's a breakdown of the key components:

A. Imports

a) **BlynkLib**: Library for interacting with the Blynk platform.
b) **RPi.GPIO**: Library for controlling GPIO pins on the Raspberry Pi.
c) **BlynkTimer**: A timer class for scheduling tasks with Blynk.

B. Constants and Setup

a) **BLYNK_AUTH_TOKEN**: A unique token used for authenticating the device with Blynk.
b) **led1 and led2**: Variables representing GPIO pin numbers (23 and 24) where the LEDs are connected.
c) **GPIO.setmode(GPIO.BCM)**: Sets the GPIO pin numbering to BCM (Broadcom SOC channel).

d) **GPIO.setup()**: Configures the specified GPIO pins as output.

C. Blynk Initialization

a) The Blynk instance is initialized with the authentication token, allowing communication with the Blynk server.

D. Virtual Pin Handlers

a) **@blynk.on("V0")**: This function handles input from virtual pin V0. When a value is received, it checks if the value is not zero:
   I. If true, it turns LED1 on (`GPIO.output(led1, GPIO.HIGH)`) and prints "LED1 HIGH".
   II. If false, it turns LED1 off and prints "LED1 LOW".

b) **@blynk.on("V1")**: Similar to the V0 handler, this function controls LED2 using virtual pin V1, following the same logic.

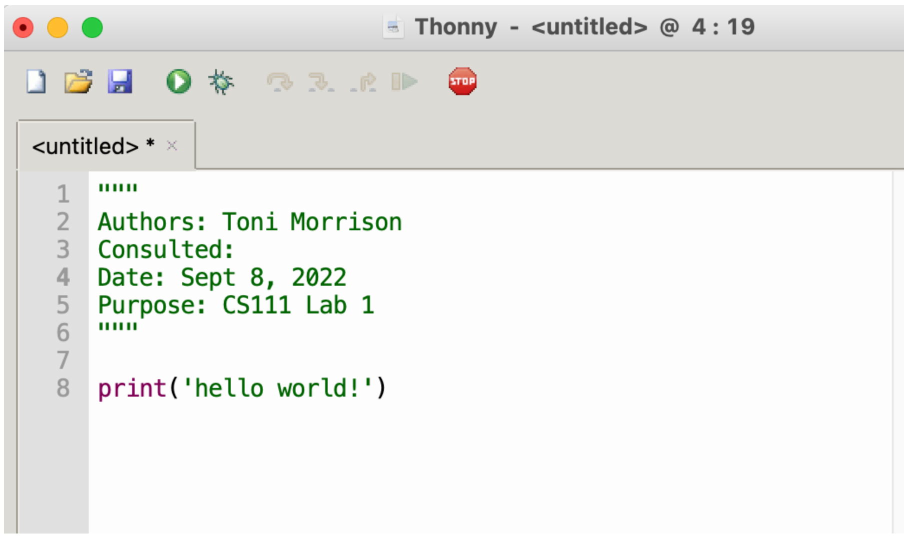
E. Key Functionality

a) The code allows remote control of two LEDs via the Blynk app, where users can toggle the LEDs by sending commands to the virtual pins V0 and V1.

Working Procedure:

1. First turn on the power switch on the ES-101 module. After the module is turned on, the PuTTY software should be turned on from the PC and communication between the PuTTY software and VNC software should be established as per Lab Manual 1(A).

2. Install the “gpiozero” Library** (if not already installed):
   Open a terminal on your Raspberry Pi and install the “gpiozero” library with:
   sh
   ```
   pip3 install blynk-library
   ```
3. Create a Python script to make the Blynk LED control . Open a text editor on your ES-101 and create a file named “blynk_led.py”.
4. Execute the script by pressing RUN Button



```
"""
Authors: Toni Morrison
Consulted:
Date: Sept 8, 2022
Purpose: CS111 Lab 1
"""

print('hello world!')
```

5. Now Press LED1 Switch in Blynk App
6. when LED1 Switch is press the LEDs 1 will turn on again when LED1 Switch is press LED 1 is turn off
7. Press Ctrl + C on the computer keyboard to close the program. After that write exit () command and press enter button.

Reference Book

1. Raspberry Pi Cookbook: Software and Hardware Problems and Solutions by Simon Monk
2. Python Programming for Raspberry Pi, Sams Teach Yourself in 24 Hours by Richard Blum & Christine Bresnahan