



# **BANGABANDHU SHEIKH MUJIBUR RAHMAN AVIATION AND AEROSPACE UNIVERSITY (BSMRAAU)**

## **Real Time Embedded System Sessional**

---

**Experiment No. 04: Understand the configuration of BLYNK Server and IoT based Real time Sensor data monitoring system design.**

**Objectives:** The objectives of this experiment are-

- a) Learn how to use Blynk Server
- b) To understand the real time sensor data upload and monitoring system
- c) To understand the IoT concept

**Equipment:**

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

**Theory:**

The SHT20 sensor is a highly accurate digital temperature and humidity sensor that communicates via the I2C interface, making it an excellent choice for various IoT applications. When integrated with a Raspberry Pi, this sensor allows for real-time environmental monitoring and data collection. The Raspberry Pi, a versatile single-board computer, can easily interface with the SHT20 and handle network communication. By utilizing the Blynk server, users can send the sensor data to a cloud-based platform, enabling remote access and control through a user-friendly mobile app. This combination of the SHT20, Raspberry Pi, and Blynk provides a powerful solution for creating smart home systems, environmental monitoring stations, and educational projects, facilitating efficient data visualization and management from anywhere in the world.

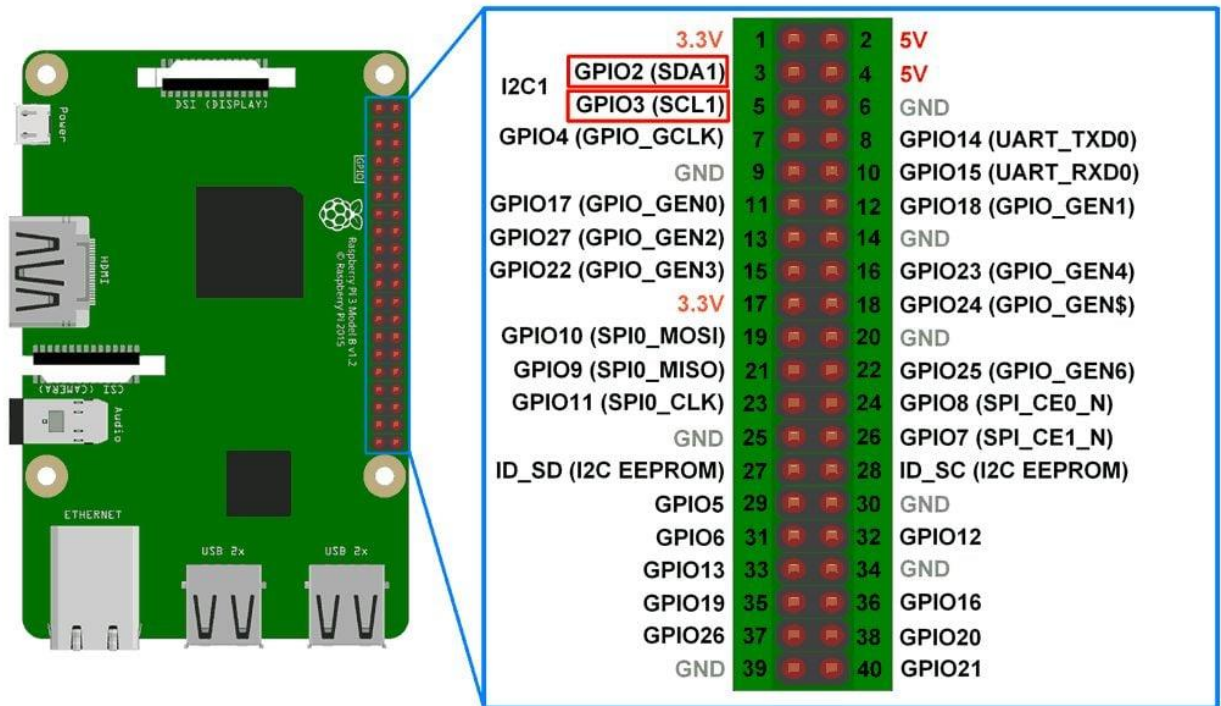


Figure:1

## 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. SHT20 LIBRARY

The `SHT20` library is typically used for interfacing with the SHT20 temperature and humidity sensor. This sensor is popular in various applications due to its accuracy and reliability. Here's a brief overview of its features and usage:

### Key Features:

- **Temperature Measurement:** Provides accurate temperature readings, usually ranging from -40°C to 125°C.
- **Humidity Measurement:** Measures relative humidity from 0% to 100% with high precision.
- **I2C Interface:** Communicates over the I2C bus, making it easy to connect to microcontrollers like Arduino or Raspberry Pi.

### Basic Usage:

1. **Installation:** Make sure you have the necessary library installed. You can often find it in the Arduino Library Manager or on platforms like GitHub.
2. **Connecting the Sensor:** Wire the SHT20 sensor to your microcontroller according to the I2C specifications (SDA to SDA, SCL to SCL, etc.).
3. **Sample Code:**

```
Python
from sht20 import SHT20
# Create an instance of the SHT20 sensor
sensor = SHT20()
# Read temperature and humidity
temperature = sensor.read_temperature()
humidity = sensor.read_humidity()

print(f"Temperature: {temperature} °C")
print(f"Humidity: {humidity} %")
```

**D. 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.

### Code:

```
import BlynkLib

from BlynkTimer import BlynkTimer

from sht20 import SHT20

import time

sht = SHT20(1, resolution=SHT20.TEMP_RES_14bit)

BLYNK_AUTH_TOKEN = 'g8GRMDXA15EMIAmKMBfXE3tyFUvVliJx'

# Initialize Blynk

blynk = BlynkLib.Blynk(BLYNK_AUTH_TOKEN)

# Create BlynkTimer Instance

timer = BlynkTimer()
```

```

# function to sync the data from virtual pins

@blynk.on("connected")
def blynk_connected():
    print("Hi, You have Connected to New Blynk2.0")
    print(".....")
    time.sleep(2);

# Function for collect data from sensor & send it to Server
def myData():
    humidity = sht.read_humid()
    temperature = sht.read_temp()
    if humidity is not None and temperature is not None:
        print("Temp={0:0.1f}C Humidity={1:0.1f}%".format(temperature, humidity))
    else:
        print("Sensor failure. Check wiring.");
    blynk.virtual_write(0, humidity,)
    blynk.virtual_write(1, temperature)
    print("Values sent to New Blynk Server!")
timer.set_interval(2, myData)

while True:
    blynk.run()
    timer.run()

```

## Code Explanation:

### A. Imports

```

Python
import BlynkLib
from BlynkTimer import BlynkTimer
from sht20 import SHT20
import time

```

- a) **BlynkLib**: Library to interface with the Blynk IoT platform.

- b) **BlynkTimer**: Used for scheduling tasks at regular intervals.
- c) **SHT20**: Library for the SHT20 sensor.
- d) **time**: Standard library for handling time-related tasks.

## B. Sensor Initialization

```
Python
sht = SHT20(1, resolution=SHT20.TEMP_RES_14bit)
```

- a) Initializes the SHT20 sensor on I2C bus 1 with a 14-bit temperature resolution.

## C. Blynk Initialization

```
Python
BLYNK_AUTH_TOKEN = 'g8GRMDXA15EMlAmKMBfXE3tyFUvVliJx'
```

- a) This is the authentication token for your Blynk project. Replace it with your own token for your project.

```
Python
blynk = BlynkLib.Blynk(BLYNK_AUTH_TOKEN)
```

- b) Creates a Blynk instance using the provided authentication token.

## D. Blynk Connection Handler

```
Python
@blynk.on("connected")
def blynk_connected():
    print("Hi, You have Connected to New Blynk2.0")
    print(".....")
    time.sleep(2)
```

- a) This function runs when the Blynk connection is established, printing a confirmation message.

## E. Data Collection and Sending Function

```
Python
def myData():
    humidity = sht.read_humid()
    temperature = sht.read_temp()
    if humidity is not None and temperature is not None:
        print("Temp={0:0.1f}C Humidity={1:0.1f}%".format(temperature, humidity))
    else:
        print("Sensor failure. Check wiring.")
```

```
blynk.virtual_write(0, humidity)
blynk.virtual_write(1, temperature)
print("Values sent to New Blynk Server!")
```

- a) This function reads temperature and humidity from the SHT20 sensor.
- b) If the readings are valid, it prints them and sends the data to Blynk virtual pins 0 and 1.
- c) If there's a sensor failure, it prints an error message.

## F. Setting Up the Timer

```
Python
timer.set_interval(2, myData)
```

- a) This schedules the myData function to run every 2 seconds.

## E. Main Loop

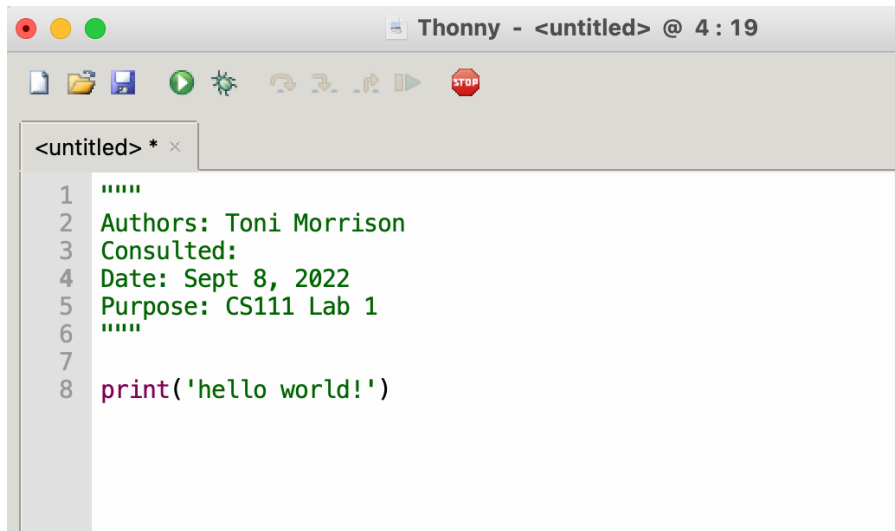
```
Python
while True:
    blynk.run()
    timer.run()
```

- a) This loop continuously runs the Blynk library and the timer to ensure data is sent at the scheduled intervals..

## 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 Temp Data . Open a text editor on your ES-101 and create a file named “blynk\_temp.py”.
4. Execute the script by pressing RUN Button



The image shows a screenshot of the Thonny Python IDE. The window title is "Thonny - <untitled> @ 4 : 19". The toolbar includes icons for file operations, running, and stopping. The editor shows a script with the following content:

```
1  """
2  Authors: Toni Morrison
3  Consulted:
4  Date: Sept 8, 2022
5  Purpose: CS111 Lab 1
6  """
7
8  print('hello world!')
```

5. Now Open Blynk App and you can see the humidity and temperature.
6. 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