Journal of Multidisciplinary Engineering Science and Technology (JMEST)
ISSN: 2458-9403
Vol. 6 Issue 12, December - 2019

Distance Control of Water Temperature via Android Devices

M. Papoutsidakis
Dept. of Industrial Design and Production Engineering
University of West Attica
Athens, Greece
mipapou@uniwa.gr

A.Chatzopoulos
Dept. of Industrial Design and Production Engineering
University of West Attica
Athens, Greece

D. Piromalis
Dept. of Industrial Design and Production Engineering
University of West Attica
Athens, Greece

***Abstract*— In this thesis we will describe the wireless communication of a microcontroller with internet connection capabilities and the reception of remote data such as liquid temperature from waterproof temperature sensors that are properly connected to the microcontroller and provide information on the precise temperature of the liquid located and then fully display real-time the data on authorized Android devices in a graphical environment specially designed for the application. It also informs the user visually through the water temperature device about the temperature value and alerts them if the temperature value exceeds the prescribed limits. This thesis titled: " WATER TEMPERATURE DISTANCE CONTROL SYSTEM VIA ANDROID DEVICES " was compiled entirely by Gkoutsioudis Emmanouil, a student in the Department of Industrial Design and Production Engineering. This particular book, as well as the implementation of the remote water temperature monitoring system via Android devices, were completed in October 2019, under the supervision of Assistant Professor Michael Papoutsidakis.**

Keywords—component; formatting; style; styling; insert (key words)

I. INTRODUCTION

IoT: In recent years, due to the constant evolution of technology and the public's need for greater autonomy, more and more intelligent, automated and autonomous devices have been developed with the capability of controlling, receiving and communicating with each other in real time remotely via the local network or the world wide web. These smart devices have expanded the horizons of the technological world into a new category of smart devices, which have the capability to interconnect and exchange data and enable the user to remotely control any device connected to the world web or local area network. The communication network and how these smart devices are interconnected with the capabilities mentioned above is defined as IoT (Internet of Things).

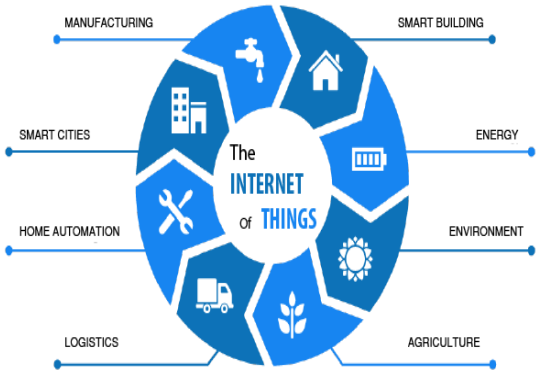


Fig. 1. *Internet of things*

NodeMCU: NodeMCU is an open source development platform designed to easily deploy IoT devices. It includes the appropriate hardware/software in collaboration with the highly economical ESP8266 WiFi SoC (System-on-a-chip) microprocessor designed and manufactured for years by Espressif Systems and hardware based on the ESP-12E module. The ESP8266 contains all the elements of a modern microcontroller such as: Tensilica Xtensa® 32-bit LX106 central processing unit (CPU) operating at 80 to 160 MHz clock adjustable clock, 128kB internal random access memory (RAM), Flash memory storage of 4MB programs and data, Wi-Fi 802.11b / g / n, so that it cannot only connect to a Wi-Fi network and interact through the Internet, but also create its own wireless network, allowing other devices to connect directly to it. Also, the ESP8266 is based on a modern operating system with full interconnection with TCP / IP protocols and SDKs. So, the ESP8266 allows microcontrollers to connect to the Internet over a WiFi network and make simple TCP / IP connections. ESP8266 was integrated with all of its features into a deployment platform called NodeMCU. The NodeMCU deployment platform features: a built-in Micro-USB port for direct power supply to the 5V DC device and for easy and immediate programming of the ESP8266 microcontroller through the converter including the CP2102 USB to UART bridge controller which converts the signal USB in serial format and allows communication between our computer and the ESP8266 and subsequent programming, reset button,

flash button for download of new software upgrade programs, 2.4GHz WiFi antenna, built-in voltage stabilizer to keep voltage constant at 3.3V as ESP8266 operating voltage is 3V to 3.6V DC, LED lights for providing user information, one of which is programmable in blue, and standard-sized GPIO (General-Purpose-Input-Output) pins that allow NodeMCU to be easily mounted on all interface boards, and breadboard type test boards. NodeMCU provides a variety of capabilities to the user through the pin-out pins of the deployment board. Specifically, it allows the user two-way communication with compatible external devices, electronic devices and sensors to input or export data through GPIO universal inputs / outputs. The D (0, 1, 2...8) GPIOs of the NodeMCU deployment platform are digital inputs / outputs and all readable PWM (Pulse-Width-Modulation) signals except D0. It also provides the ability to read analog input from external devices via pin A0 (ADC). The ADC is a 10-bit converter that converts the analog signal into digital form. The NodeMCU deployment platform is powered via the Vin pin or as mentioned above via the Micro-USB port with 5V DC. The NodeMCU 3V3 output pins provide the user with approximately 3.3V DC for general use via the voltage stabilizer. The maximum output current that NodeMCU pins such as GPIO and 3V3 can handle is up to 12mA to avoid any damage. Below is a drawing showing all the pins of the NodeMCU development platform for a better understanding.

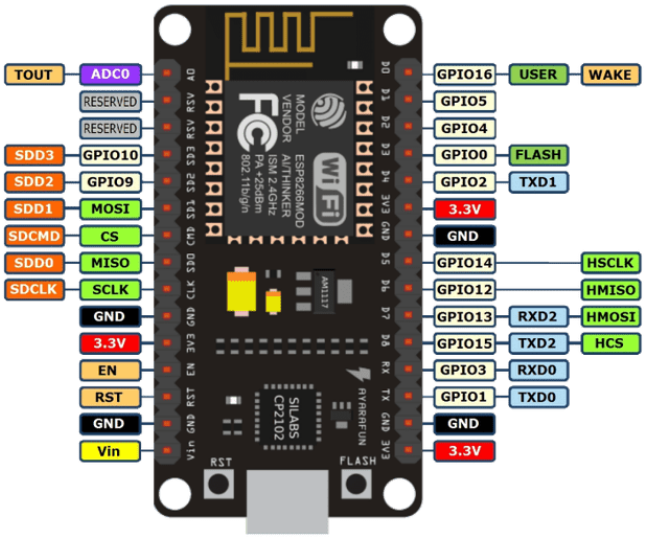


Fig. 2. *Nodemcu pin-out diagram*

DS18B20 waterproof digital temperature sensor: The DS18B20 is a waterproof digital temperature sensor that allows us to accurately measure the temperature in liquids. The temperature range that can be measured is from -55 °C to + 125 °C to within ± 0.5 °C (-10 °C to + 85 °C). The DS18B20 digital temperature sensor is compatible with most development boards on the market and features a unique 64-bit serial number, thereby enabling multiple sensors to be connected together on the same bus cable for multiple temperature measurement. The operating voltage of the DS18B20 digital temperature sensor ranges from 3.0V to 5.5V DC. The DS18B20 digital temperature sensor and its connection to a compatible microcontroller require a 4.7Kohm resistor for function properly. The data is sent from the DS18B20 digital temperature sensor via a 1-Wire interface to the Dallas 1-wire communication protocol, so in addition to the ground cable, only one cable needs to be connected to the microcontroller for reading, writing data, and the temperature conversions are output from the same data line without the need for an external power source.

Fig. 3. *DS18B20 pin-out*

RGB Light Emitting Diode common cathode: This RGB light emitting diode common cathode consists of four terminals, one terminal for red, one terminal for green, one terminal for blue and one terminal which is the common threshold for light transmittance. For the implementation of the electronic circuit, three resistors of 51ohm were arranged in series with each terminal color, to control the amount of current passing through the light emitting diode.

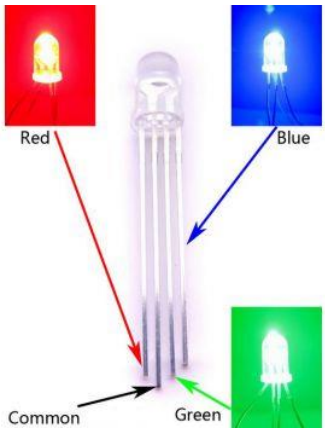


Fig. 4. *RGB light emitting diode*

II. METHODOLOGY

Blynk App: An IoT platform called Blynk was used to implement and design the application for receiving data remotely via Android devices. Blynk is a relatively new online platform that allows you to easily and quickly design the interface between a Blynk compatible microcontroller and an authorized Android device after downloading the specific application to the Android device through the Play Store. Blynk platform-compatible microcontrollers are more than 400, among them the most widely used such as various versions of Arduino, Raspberry Pi, ESP8266, Wemos D1, Intel Galileo, NodeMCU and others. A basic prerequisite for downloading the Blynk app on Android devices from

the Play Store is to have Android OS 4.2 and higher. The Blynk app is designed and compatible with iOS devices as well, provided that they have an iOS 9 operating system or higher and the app is downloaded through the App Store.

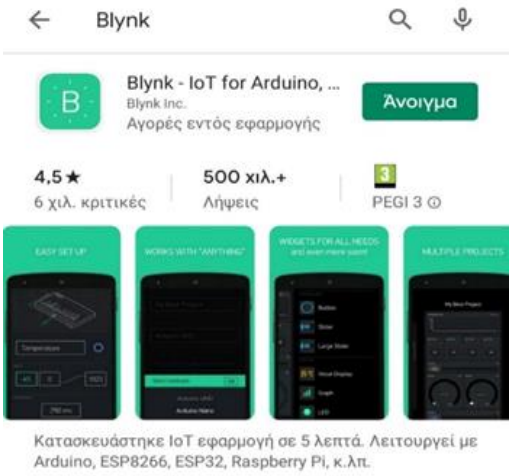


Fig. 5. *Download the Blynk App through the Play Store*

After downloading the Blynk app through the Play Store on a compatible Android device, you need to set up an account in the app and verify our details through the email account we registered at registration.Once the account has been successfully created for the implementation of our project in the Blynk application remains as follows:

- Creating a new project
- Enter a project name
- Selecting a compatible device for project implementation and sharing
- Data (Arduino, Raspberry, NodeMCU)
- Select connection type (WiFi, Ethernet, Bluetooth, BLE, GSM, USB)
- Select theme (Dark or Light)

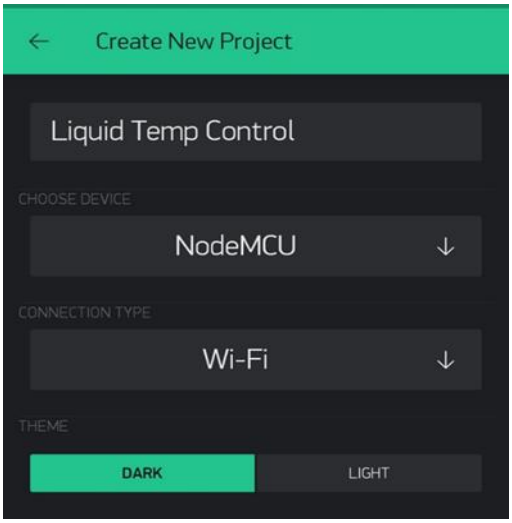
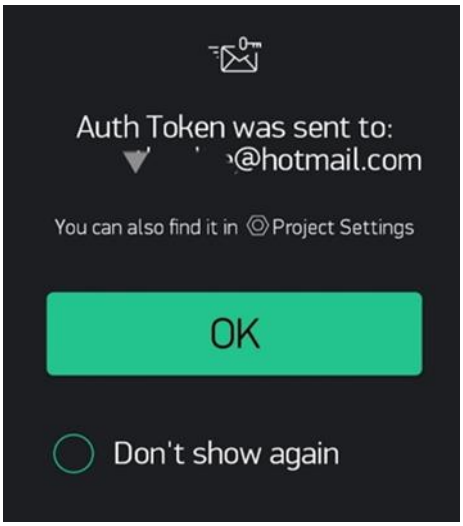


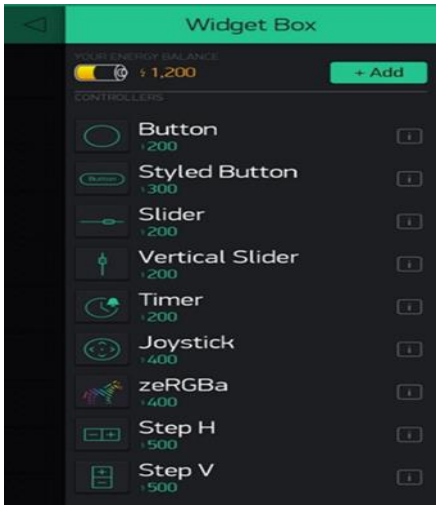
Fig. 6. *Create a new project in the Blynk App*

Once we have made the right choice based on our project implementation needs, it is automatically sent by Blynk to the email account that we have declared an authentication token. The authentication token is a 32-character set, a unique identifier required to connect the hardware to our smartphone and which we must include in the code for hardware and application synchronization. Auth Token is not shared with anyone unless we want someone to access our hardware.



The next step is to give our project the proper shape through the available widgets provided by Blynk. By tapping on the addition symbol or anywhere on the canvas we can choose from a range of functional graphics.

Blynk's application through the functional widgets it provides us enormous potential for realizing our project, graphically displaying all the data we receive from our various inputs, and storing data in Blynk's cloud, remote control of our device only by authorized control devices, receive device status notifications and use the already existing sensors of the controller device.



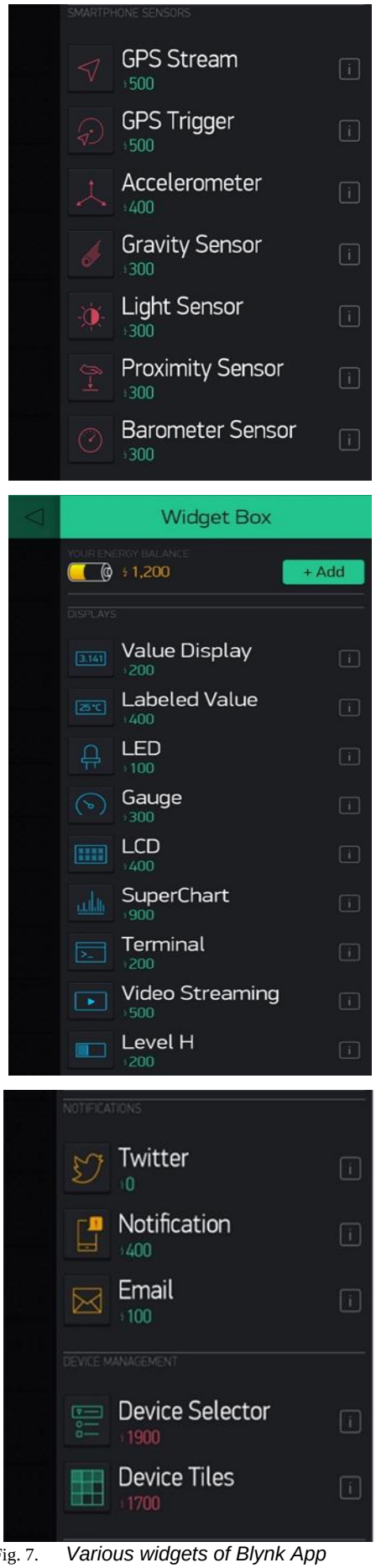


Fig. 7. *Various widgets of Blynk App*

For the implementation of this project and for the visual inspection of the water temperature remotely, the graphical interface was appropriately designed and the following graphics were used and adapted:

- Gauge
- Labeled Value
- Level H

To receive alerts and messages on the Android device if the water temperature exceeds the limits (water temperature <10 °C and water temperature> 50 °C) the following graphic Blynk widget was properly used and programmed:

- Notification

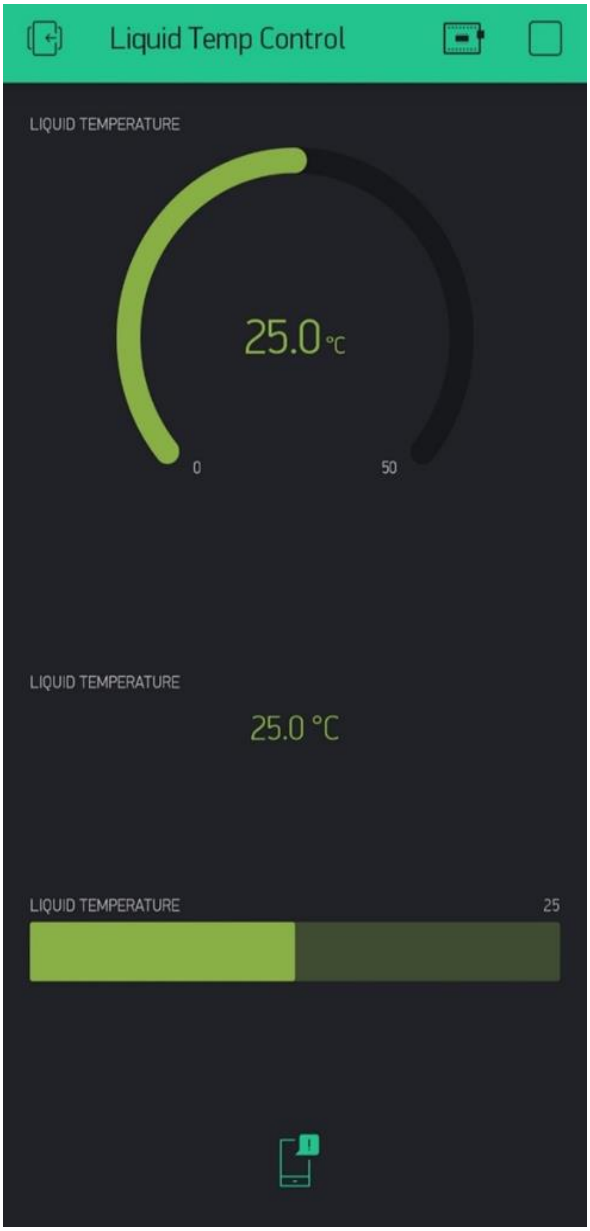


Fig. 8. *Graphic interface*

Arduino IDE: Arduino IDE was used to program and synchronize the NodeMCU microcontroller with the Blynk application platform, for data exchange and remote control of the water temperature control

system. The Arduino IDE (Integrated Development Environment) is a user-friendly programming environment used to write and send code to the Arduino family of controllers and beyond. The Arduino IDE programming language is based on C / C ++ programming languages with slight variations in the syntax of some commands.

After installing the Arduino IDE on our PC, programming and synchronizing the NodeMCU microcontroller with the Blynk application platform will require the following steps:

Open the Arduino IDE

File-> Create a new project

File-> Preferences -> Additional Board Administrator URLs-> and we add ***"http://arduino.esp8266.com/stable/package_esp8266com_index.json"***

Tools-> Board-> Board Manager -> we search for NodeMCU board and after finding the right version we download it

Tools-> Library Management-> we search and install the Blynk library

And then Coding...

```
// Libraries
#define BLYNK_PRINT Serial         // Blynk Library
#include <ESP8266WiFi.h>          //  Esp8266 Library
#include <BlynkSimpleEsp8266.h>  //Blynk_Esp8266 Library
#include <OneWire.h>             //OneWire Library for Ds18b20
#include <DallasTemperature.h> // Dallas Library for Ds18b20
#include <SimpleTimer.h>       //  Timer Library

OneWire oneWire(D2);        // Declaration Digital Pin D2 of NodeMCU
DallasTemperature sensors(&oneWire); //  Declaration sensor
SimpleTimer timer;                   //   Declaration Timer


// Declarations of Variables
char auth[] = "********";    // Declaration of Auth Token Blynk App
char ssid[] = "********";   //  Declaration of WiFi Credentials
char pass[] = "********"; //   Declaration of WiFi Credentials
float  temp = 0;          //    Declaration of Temperature Value
float SendNotification = 0;  // Declaration of Notification
int mem= 0;                  //  Declaration of Memory
float brightvalue= 0;      // Declaration of Brightness Value RGB LED
```

III. Conclusion

This remote water temperature monitoring system through Android devices was designed and built to optimize and simplify our simple daily procedures, such as water temperature control. At a time when almost everyone has internet access and with some money, we can get an Android device if we don't already have it, our remote water temperature monitoring system can alert us anytime on the other side of the earth about the exact temperature of the water and not only.

Acknowledgment

All authors would like to thank the University of West Attica and specifically the Post Graduate Program of Studies (MSc) "New Technologies in Shipping and Transportations", for the financial support provided to them to undertake this research project.

References

[1] https://el.wikipedia.org/wiki/

[2] https://www.electronicshub.org/microcontrollers-basics-structure-applications/

[3] https://www.elprocus.com/microcontrollers-types-and-applications/

[4] https://en.wikipedia.org/wiki/Perfboard

[5] https://rfnews.gr/?p=4634

[6] https://el.wikipedia.org/wiki/

[7] https://el.wikipedia.org/wiki/

[8] https://en.wikipedia.org/wiki/NodeMCU

[9] https://en.wikipedia.org/wiki/ESP8266

[10] https://developer.ibm.com/tutorials/iot-nodemcu-open-why-use/

[11] https://lastminuteengineers.com/esp8266-nodemcu-arduino-tutorial/

[12] https://www.racksolutions.com/news/data-center-news/top-10-largest-data-centers-world/

[13] https://blynk.io/