**VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY**

**UNIVERSITY OF INFORMATION TECHNOLOGY**

**FACULTY OF COMPUTER ENGINEERING**

**Cao Phan Tien Dung - 19521387 Nguyen Van Tin - 19521387**

**Ngo Man Dat - 19521333**

**MIDTERM PROJECT SMART GARDEN**

**CE224.M13.MTCL(EN)**

**HO CHI MINH CITY, 11/2021**

**VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY UNIVERSITY OF INFORMATION TECHNOLOGY**

**FACULTY OF COMPUTER ENGINEERING**

**CAO PHAN TIEN DUNG – 19521387**

**NGUYEN VAN TIN**

**– 19521022**

**NGO MAN DAT**

**- 19521333**

**MIDTERM PROJECT**

**SMART GARDEN**

**CE224.M13.MTCL(EN)**

**MENTOR PhD. TRI NHUT DO**

**HO CHI MINH CITY, 11/2021**

|  |  |
| --- | --- |
| VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY  **UNIVERSITY OF INFORMATION**  **TECHNOLOGY** | **SOCIALIST REPUBLIC OF VIETNAM**  **Independence – Freedom - Happiness** |

**DETAILED TOPICS**

|  |
| --- |
| **VIETNAMESE PROJECT NAME:** Khu vườn thông minh |
| **ENGLISH PROJECT NAME:** SMART GARDEN |
| **Instructor** PhD. TRI NHUT DO, Department of Computer Engineering |
| **Implementation time:** From: 08/10/2021 To: 19/11/2021 |
| **Student Perform:**  CAO PHAN TIEN DUNG – 19521387 NGUYEN VAN TIN 19521022  NGO MAN DAT - 19521333 |
| **Overview of the topic:** The project proposes to build a smart garden system with a small scale. With the purpose of studying and executing intelligent system design.  **The goal of the project:** System can collect data of the garden and this data will be stored and displayed in web app or mobile app. Besides, the system can work well in most of weather conditions.  **Main content of the topic:**  Programming language (s): C / C ++. (arduino) |

|  |  |
| --- | --- |
| Sub-system (s): Blynk, Thinkspeak IoT,  Hardware: Esp8266 and sensor | |
|  | |
| **Certification of Instructor**  (Sign and clearly state full name) | **HCM city, 2021 November 19 Student**  (Sign and clearly state full name) |

**TABLE OF CONTENT**

[Chapter 1. INTRODUCTION 7](#_Toc88165236)

[Chapter 2. BUILDING MATERIALS 8](#_Toc88165237)

[**2.1. List of material 8**](#_Toc88165238)

[**2.2. Diagram of materrials 8**](#_Toc88165239)

[Chapter 3. FUNCTIONING DESCRIPTION 12](#_Toc88165240)

[**3.1. System diagram 12**](#_Toc88165241)

[Chapter 4. FLOW CHAR 13](#_Toc88165242)

[Chapter 5. SOFTWARE DESCRIPTION 14](#_Toc88165243)

[**5.1. Arduino 14**](#_Toc88165244)

[**5.2. Source code 15**](#_Toc88165245)

[Chapter 6: SUB-SYSTEM PART 26](#_Toc88165246)

[1. ThingSpeak Key Capabilities 29](#_Toc88165247)

[2. Connect Your Hardware to ThingSpeak 29](#_Toc88165248)

[3. Access Your Data Both Online and Offline 30](#_Toc88165249)

[4. Remotely Visualize Sensor Data in Real Time 32](#_Toc88165250)

[**5. Control Devices Online with One Second Update Rates 32**](#_Toc88165251)

[**6. Perform Computations and Build Custom Visualizations 34**](#_Toc88165252)

[**7. Create Streaming Analytics, and Integrate with Your Systems 34**](#_Toc88165253)

**PROJECT SUMMARY**

This project proposes to build a smart garden which receives data from sensor and sends data to Thingspeak IoT server and receives the control signals from Blynk app .

This system will collect humidity, temperature, soil moisture, motor status data.

# Chapter 1. INTRODUCTION

Our team's project includes a central controller and sensors to collect data from the environment. Devices display via hardware, phone as well as via web server.

With this project, we hope to promote the field of research on smart agricultural systems in the area in the particular as well as in the general of Vietnam.

Our system operates as an intelligent assistant that can help farmers observe their garden.

This assistant looks over the garden through sensors and report to the phone and web app. Then, farmers can know the conditions of their garden and make appropriate decisions.

According to the research process of our team, the most suitable soil moisture is in the range of 60 percent to 70 percent. Therefore, our device also has a smart state that the humidity below the allowable level will automatically control the irrigation device, and the soil moisture content is above the maximum allowable level, the irrigation device will must turn off.

As a result of what I have learned, I hope the system can serve farmers in particular and Vietnamese agriculture in general.

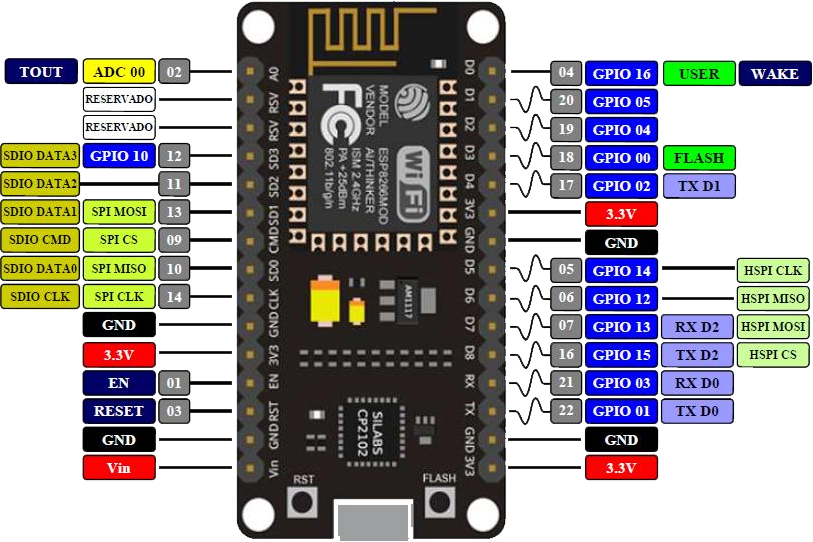
# Chapter 2. BUILDING MATERIALS

## 2.1. List of material

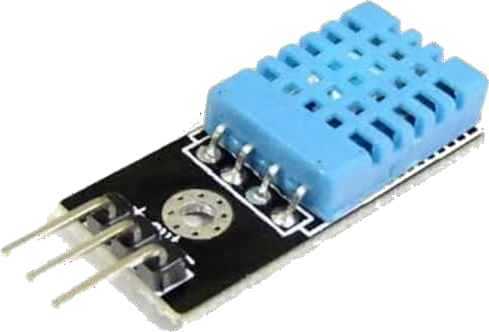
|  |  |
| --- | --- |
| - | ESP8266 NODEMCU |
| - | Sensor DHT11 |
| - | Sensor Soil Moisture |
| - | Sensor Touch TTP223 |
| - | Button |
| - | LCD 16x2 I2C |
| - | Motor 12V DC |
| - | Led |

## 2.2. Diagram of materrials

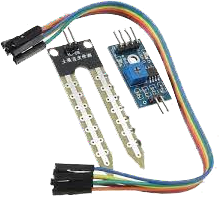
- **ESP8266 NODEMCU**



* **Sensor DHT11**



* **Sensor Soil Moisture-**



* **Sensor Touch TTP223**



* **Button**



* **LCD 16x2 I2C**



* **Motor 12V DC**

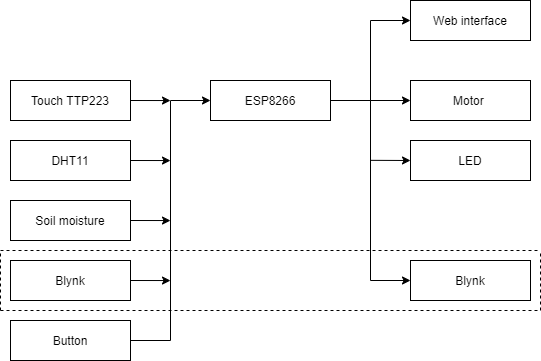


* **LED**

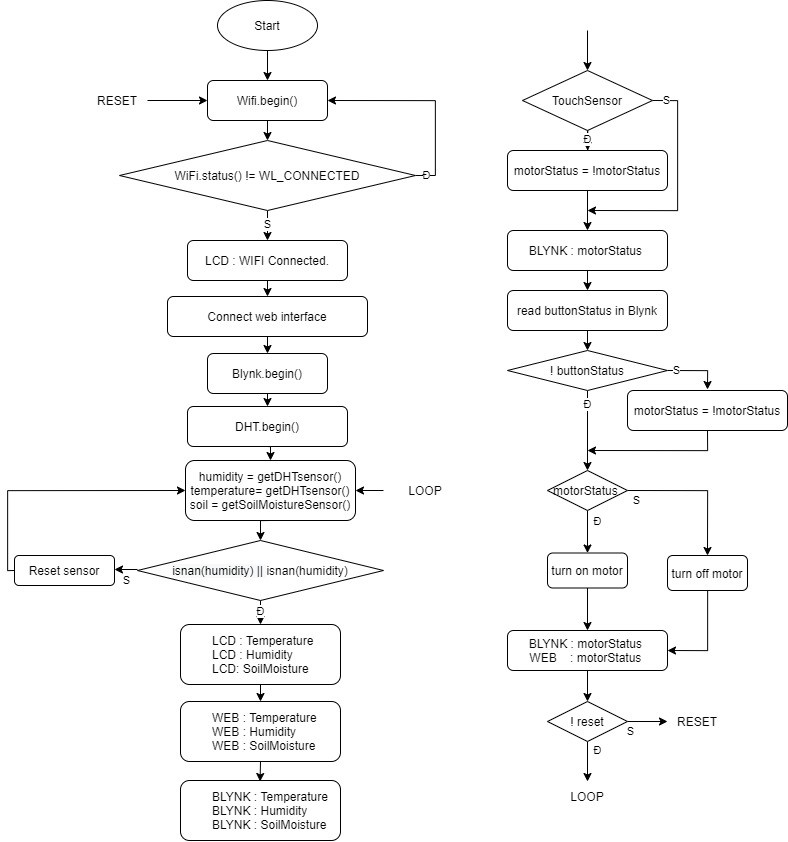


# Chapter 3. FUNCTIONING DESCRIPTION

## 3.1. System diagram



# Chapter 4. FLOW CHAR



# Chapter 5. SOFTWARE DESCRIPTION

## 5.1. Arduino

Arduino IDE is an open-source software, designed by Arduino.cc and mainly used for writing, compiling & uploading code to almost all Arduino Modules. It is an official Arduino software, making code compilation too easy that even a common person with no prior technical knowledge can get their feet wet with the learning process. It is available for all operating systems i.e., MAC, Windows, Linux and runs on the Java Platform that comes with inbuilt functions and commands that play a vital role in debugging, editing and compiling the code. A range of Arduino modules available including Arduino Uno, Arduino Mega, Arduino Leonardo, Arduino Micro and many more. The main code, also known as a sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded in the controller on the board. The IDE environment mainly contains two basic parts: Editor and Compiler where former is used for writing the required code and later is used for compiling and uploading the code into the given Arduino Module. The code is written by C and C++ language

## 5.2. Source code

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | /\*  KHAI BÁO PIN  \*/  #define LED 2 //D4 led nguồn #define MOTOR 13 //D7 #define DHTPIN 12 //D6 #define DOAMDAT A0 //analog #define TOUCH 16 //D0  //#define SDA 5 //D1  //#define SCL 4 //D2  /\*  KHAI BÁO THƯ VIỆN  \*/  /\*---------HIỂN THỊ \*/  //#include <Wire.h>  #include <LiquidCrystal\_I2C.h>  /\*---------KẾT NỐI \*/  #include <ESP8266WiFi.h> #include <ESP8266HTTPClient.h> #include <WiFiClient.h>  #include <BlynkSimpleEsp8266.h>  //web server  #include <ThingSpeak.h>  /\*---------CẢM BIẾN \*/  #include <DHT.h> #include <DHT\_U.h> |

|  |  |
| --- | --- |
| 31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60 | /\*  KHAI BÁO BIẾN TOÀN CỤC  \*/  //BLYNK  //#define BLYNK\_TEMPLATE\_ID "TMPLxxxxxx" #define BLYNK\_DEVICE\_NAME "My\_System"  #define BLYNK\_AUTH\_TOKEN  "a64YarnJsGcQ\_jrmC1L0jJARj84M7u3y" #define BLYNK\_PRINT Serial  char auth[] = BLYNK\_AUTH\_TOKEN;  //THINKSPEAK  const char\* server = "api.thingspeak.com"; unsigned long myChannelNumber = 1433980;  const char \* myWriteAPIKey = "0B2TBIQVLGHX2V5X"; const char \* myReadAPIKey = "C89KRRWFFEIGBWOE";  WiFiClient client;  char ssid[] = "FPT Ngoc Nhan"; char pass[] = "khongnoiduoc";  //char ssid[] = "ThanhTung";  //char pass[] = "984513194";  LiquidCrystal\_I2C lcd(0x27,16,2); //Khởi tạo biến lcd #define DHTTYPE DHT11 |

|  |  |
| --- | --- |
| 61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90 | DHT dht(DHTPIN, DHTTYPE); //Khởi tạo biến dht  bool motorStatus = 0; bool valTouch = 0;  /\*  HÀM KHỞI TẠO  \*/  void setup() { Serial.begin(115200);  //Debug hardware debugHardware();  //INIT LCD  lcd.init(); lcd.backlight();  //Test lcd lcd.setCursor(4,0); lcd.print("WELLCOME"); lcd.setCursor(2,1);  lcd.print("TO MY SYSTEM");  //Wifi connection WiFi.begin(ssid, pass);  while (WiFi.status() != WL\_CONNECTED)  {  delay(500); |

|  |  |
| --- | --- |
| 91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120 | Serial.print(".");  }  Serial.println(); Serial.println(WiFi.localIP());  //Thinkspeak init ThingSpeak.begin(client);  //lcd wifi connected lcd.clear(); lcd.setCursor(0,0); lcd.print("Wifi Connected"); lcd.setCursor(0,1); lcd.print(WiFi.localIP());  //init blynk  Blynk.begin(auth, ssid, pass); dht.begin();  pinMode(DOAMDAT, INPUT); pinMode(DHTPIN, INPUT); pinMode(TOUCH, INPUT); pinMode(MOTOR, OUTPUT);  /\*START\*/ digitalWrite(MOTOR, LOW); delay(1500);  }  //Đọc giá trị nút nhấn V0 //trigger |

|  |  |
| --- | --- |
| 121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150 | BLYNK\_WRITE(V0){  motorStatus = param.asInt();  }  /\*  HÀM LOOP  \*/  void loop() {  //lear lcd lcd.clear();  //run blynk Blynk.run();  //-------- Cảm biến độ Soil float Soil = getSoil();  //-------- Cảm biến độ DHT  int h = dht.readHumidity(); //Độ 'ẩm' float t = dht.readTemperature(); //Độ '`C' if(!getDHT(h,t)){return;}  //  // motorStatus = getDataBlynk();  //-------- Cảm biến touch getTouch();  //Thiết lập ngưỡng tưới tự động theo độ ẩm đất ThresholdSoilMoisture(Soil);  // Điều khiển motor |

|  |  |
| --- | --- |
| 151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180 | digitalWrite(MOTOR, motorStatus);  // Hiển thị ra lcd  lcdRun(h, t, Soil, motorStatus);  // Gửi dữ liệu lên blynk  sendDataBlynk(h, t, Soil, motorStatus);  ////Debug serial  //debugValue(h,t,Soil);  //Think speak send data to web server thinkspeakRun(h, t, Soil, motorStatus);  // Hiển thị ra lcd  lcdRun(h, t, Soil, motorStatus);  // delay(70); //Delay for control lcd  }  void debugValue(int h, float t, int Soil){  //Debug data Serial.print("Nhiet do: "); Serial.print(t); Serial.println(" `C"); Serial.print("Do am: ");  Serial.print(h); |

|  |  |
| --- | --- |
| 181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210 | Serial.println(" %"); Serial.print("Do am dat: "); Serial.print(Soil); Serial.println(" %"); Serial.println("");  }  void sendDataBlynk(int h, float t, int s, bool myMotor){ Blynk.virtualWrite(V0, myMotor); Blynk.virtualWrite(V1, t);  Blynk.virtualWrite(V2, h); Blynk.virtualWrite(V3, s);  }  void ThresholdSoilMoisture(int doamdat){  //60 - 70 is perfect  int minThreshold = 50; int maxThreshold = 90;  if(doamdat < minThreshold){motorStatus = 1;} if(doamdat >= maxThreshold){motorStatus = 0;}  }  bool getDHT(int h, float t){ if (isnan(h) || isnan(t)) {  Serial.println("Failed to read from DHT sensor!"); return 0;  }  return 1;  }  int getSoil(){ |

|  |  |
| --- | --- |
| 211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240 | //-------- Cảm biến độ ẩm đất int doam = 0;  int doamVal = 0; for(int i = 0; i<10; i++){  doam += analogRead(DOAMDAT);  }  doam = doam/10;  doamVal = map(doam, 400, 1024, 100, 0);  if(doamVal >= 100){  doamVal = 100;  }else{  if(doamVal <= 0){  doamVal = 0;  }  }  return doamVal;  }  void getTouch(){  int Touch = analogRead(TOUCH); int thresholdTouch = 100; if(Touch >= thresholdTouch){  motorStatus = !motorStatus; while(Touch >= thresholdTouch){ if(Touch < thresholdTouch){break;} Touch = analogRead(TOUCH);  }  } |

|  |  |
| --- | --- |
| 241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270 | }  void debugHardware(){  //Debug hardware pinMode(LED, OUTPUT); for(int i = 0; i<3; i++){ digitalWrite(LED, HIGH); delay(200); digitalWrite(LED, LOW); delay(200);  }  digitalWrite(LED, LOW);  }  void thinkspeakRun(int h, float t, int s, bool motorStatus){ if (client.connect(server,80)){  ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey); ThingSpeak.setField(1, t);  ThingSpeak.setField(2, h); ThingSpeak.setField(3, s); ThingSpeak.setField(4, motorStatus);  }  client.stop();  }  void lcdRun(int h, float t, int s, bool motorStatus){ lcd.clear();  lcd.setCursor(0,0); lcd.print("T:"); |

|  |  |
| --- | --- |
| 271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300 | lcd.setCursor(2,0); lcd.print(t); lcd.setCursor(6,0); lcd.print("'C H:"); if(h<10){ lcd.setCursor(13,0); lcd.print(h); lcd.setCursor(14,0); lcd.print("%");  }else{ if(h!=100){  lcd.setCursor(12,0); lcd.print(h); lcd.setCursor(14,0); lcd.print("%");  }  else{ lcd.setCursor(12,0); lcd.print("100%");  }  }  lcd.setCursor(0,1); lcd.print("RL:"); lcd.setCursor(3,1); if(motorStatus){ lcd.print("ON");  }else{ |

|  |  |
| --- | --- |
| 301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323  324  325  326  327  328 | lcd.print("OFF");  }  lcd.setCursor(6,1); lcd.print(" Humi:");  if(s<10){ lcd.setCursor(13,1); lcd.print(s); lcd.setCursor(14,1); lcd.print("%");  }else{ if(s!=100){  lcd.setCursor(12,1); lcd.print(s); lcd.setCursor(14,1); lcd.print("%");  }  else{ lcd.setCursor(12,1); lcd.print("100%");  }  }  // delay(10);  } |

### Chapter 6: SUB-SYSTEM PART

* 1. **Building application**
     1. **Blynk**

1. What is Blynk?

Blynk is developed to help users who are not very familiar with programming, simple drag and drop block interface on the phone application, flash firmware on arduino devices like UNO, Nano with ethernet shields; esp32, 8266 wifi which does not require too much knowledge - step by step just follow.

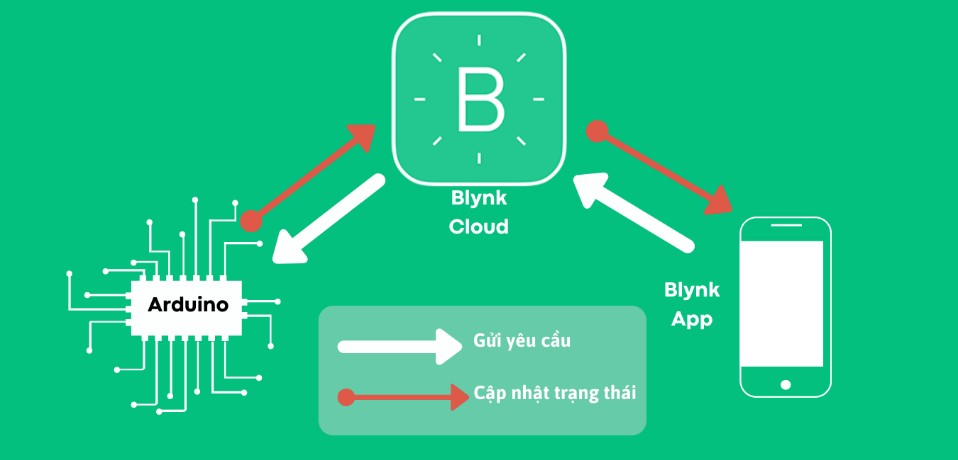
1. What can Blynk do?

Deploying a smart home system, remoting monitor, etc., the only limit is your ability as well as your creativity.

Widgets such as: push button, display value, value pull bar, draw diagram,... allow to control the arduino's GPIOs, turn on and off the relay, control home appliances.

Blynk's strength can be connected everywhere.

More specifically, you can share the control panel with other users (friends, relatives,

...) to use the same system.

1. Describe how Blynk works:

The arduino device is responsible for controlling the devices in the house via relays that plug directly into the arduino GPIOs, or any other type of communication if you can use it (RF, Uart). This device requires a network connection (ESP8266, ESP32,

...)

1. How to deploy a control system using Blynk?

If you can do it yourself, go to the link to find out. If you want a more visual description, please wait for my video. Remember to follow me to receive updates.

Basic steps:

Download Blynk app on PlayStore/App Store

Register an account, the dashboard interface will appear. Set up a console.

You can see more of my posts here.

Download the arduino library to your computer, flash it on the board, connect to the network.

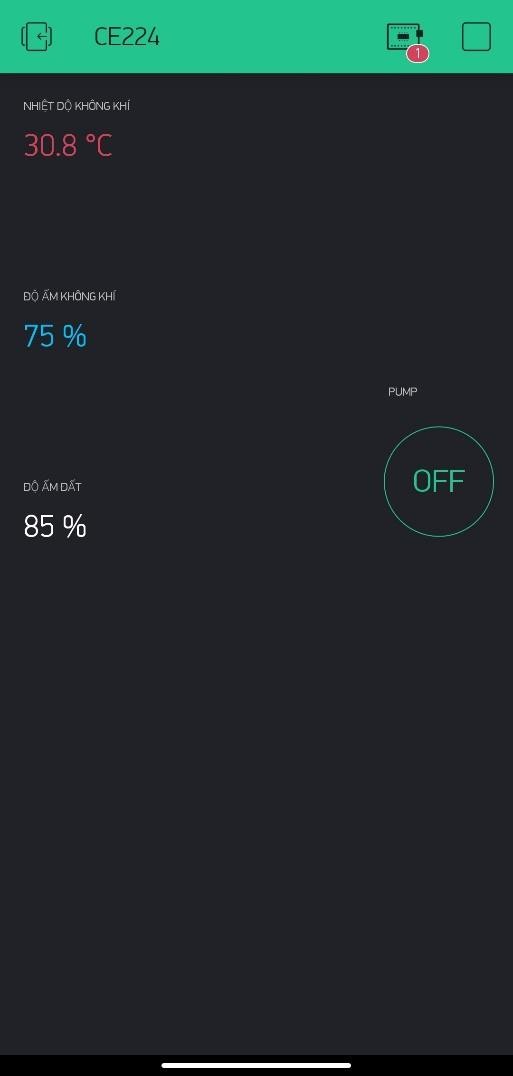
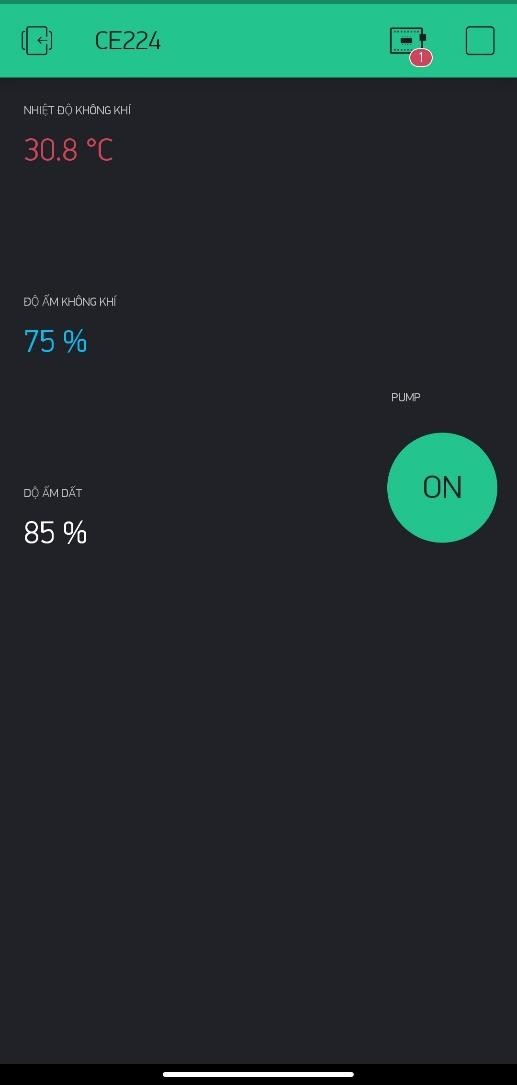
After the arduino successfully set up, App Blynk will notify you that the device is working.

1. What is the limit for Blynk?

With the free version, you can only use a limited number of widgets, each widget takes up an amount of "energy", when you bring up the main screen, you will lose this part, it like a fee. Want to put out more control page control is required to have more "energy".

There are two ways to add “energy”. You can pay extra - in app purchase. Or deploy a local server. For a small project, I think it is better to buy more energy from the supplier

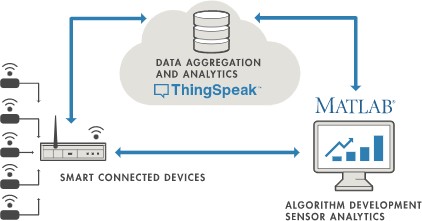
My project



* + 1. **Think speak IoT**

1. **ThingSpeak for IoT**

ThingSpeak™ is an IoT analytics platform service from MathWorks®, the makers of MATLAB® and Simulink®. ThingSpeak allows you to aggregate, visualize, and analyze live data streams in the cloud. ThingSpeak provides instant visualizations of data posted by your devices or equipment. Execute MATLAB code in ThingSpeak, and perform online analysis and processing of the data as it comes in. ThingSpeak accelerates the development of proof- of-concept IoT systems, especially those that require analytics. You can build IoT systems without setting up servers or developing web software. For small- to medium-sized IoT systems, ThingSpeak provides a hosted solution that can be used in production.



# ThingSpeak Key Capabilities

ThingSpeak allows you to aggregate, visualize and analyze live data streams in the cloud. With ThingSpeak, your data is stored in channels. Each channel stores up to 8 fields of data. You can create as many channels as you need for your application.

# Connect Your Hardware to ThingSpeak

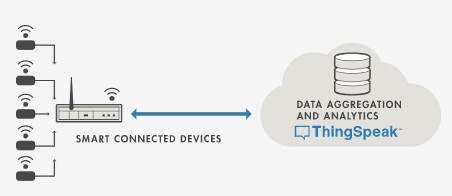
You can use any Internet-connected device with ThingSpeak. When sending data from your devices or equipment, you can use native libraries for common embedded hardware prototyping platforms like Arduino®, Espressif ESP8266 and ESP32, Particle and Raspberry Pi™. You can also send data to ThingSpeak from machines or local gateways using a [REST](https://www.mathworks.com/help/thingspeak/rest-api.html)

[API](https://www.mathworks.com/help/thingspeak/rest-api.html) or an [MQTT API.](https://www.mathworks.com/help/thingspeak/mqtt-api.html) In addition, the following vendors have built integrations to ThingSpeak to make setup even easier:

[LoRaWAN®](https://lora-alliance.org/about-lorawan/) [Things Network](https://www.mathworks.com/help/thingspeak/write_things_network.html) [Senet](https://docs.senetco.io/dev/stream/#thingspeak)

[Libelium](https://www.mathworks.com/hardware-support/libelium-sensors.html) Beckhoff [Particle devices](https://github.com/mathworks/thingspeak-particle)

If you are a Simulink user, you can use [Simulink blocks](https://www.mathworks.com/help/search.html?submitsearch&qdoc=thingspeak%2Bproduct%3Asimulink&selectedsource=mw&source=mw) in your models to write data to ThingSpeak.



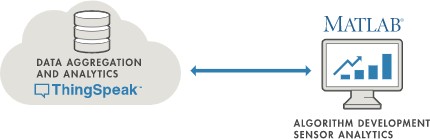
# Access Your Data Both Online and Offline

ThingSpeak stores all the information you send it in one central location in the cloud, so you can easily access your data for online or offline analysis. Your private data is protected with an API key that you control. When you are logged in to your ThingSpeak account, you can use the web to securely download the data stored in the cloud. You can also programmatically read your data in CSV or JSON formats using a [REST API](https://www.mathworks.com/help/thingspeak/rest-api.html) call and the appropriate API key.

Your devices can also read data from a ThingSpeak channel by subscribing to an MQTT topic. Import data from third-party web services including climate data from NOAA, public utility data from local utility providers, and stock and pricing data from financial providers. You can

use that data together with the data you are collecting from your devices and equipment to investigate correlations and develop predictive algorithms.

MATLAB users can import data stored in ThingSpeak into the MATLAB desktop environmen[t using the thingSpeakRead function.](https://www.mathworks.com/help/thingspeak/thingspeakread.html)





# Remotely Visualize Sensor Data in Real Time

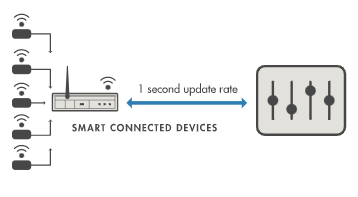
ThingSpeak automatically charts the data that you send it, so you can remotely monitor your devices or equipment from anywhere. View your data from any web browser or mobile device. Share read-only views of your data with the clients and colleagues that you specify. Alternatively, you can use ThingSpeak to manage your data, and you can build your own front end for your clients and customers to log in to.



## Control Devices Online with One Second Update Rates

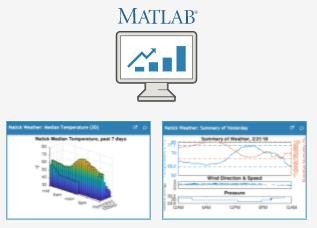
With a commercial ThingSpeak license, you can send data to ThingSpeak as fast as once every second. This not only enables near-real time monitoring of your devices, but it allows you to set up control loops from the cloud. For example, you could configure ThingSpeak to turn a light on when your motion sensor detects a person has walked into a room. For applications that require faster response times, the best practice is to have the control loop at the edge closer to the hardware.

33



## Perform Computations and Build Custom Visualizations

With the MATLAB engine built into ThingSpeak, you can perform calibrations, develop analytics, and transform your IoT data. You can also use the MATLAB engine built into ThingSpeak to build custom charts. With a commercial ThingSpeak license, you can run MATLAB calculations that last up to 60 seconds. A commercial ThingSpeak license also enables you to [use MATLAB Toolboxes](https://www.mathworks.com/help/thingspeak/matlab-toolbox-access.html) for machine learning, signal processing, system identification, and more with ThingSpeak, provided you have a license for the toolbox.

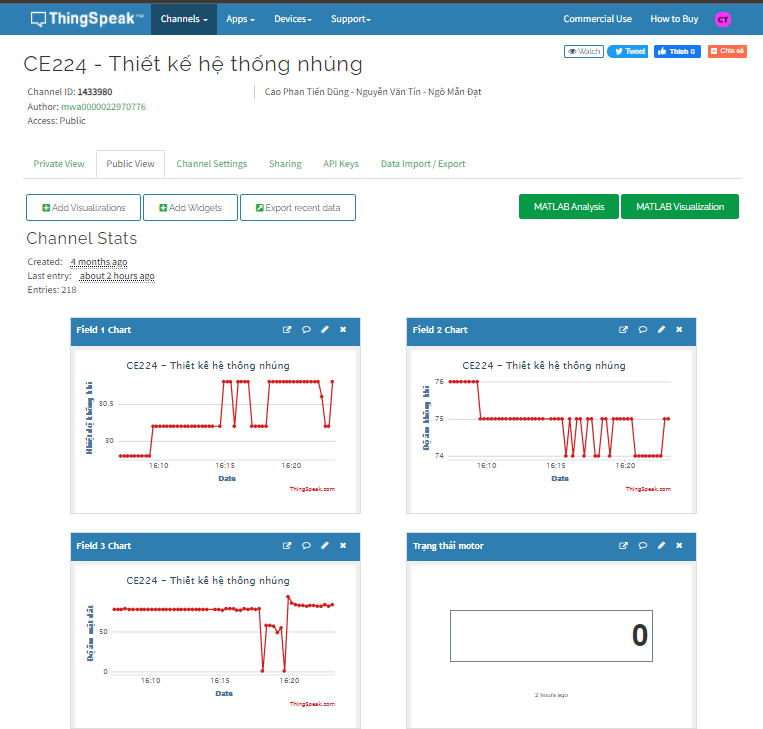


## Create Streaming Analytics, and Integrate with Your Systems

Operationalize your analytics using the Time Control and React apps. W[ith the Time Control](https://www.mathworks.com/help/thingspeak/timecontrol-app.html) [app,](https://www.mathworks.com/help/thingspeak/timecontrol-app.html) you can schedule a computation to run once a day, once an hour, or as quickly as once every 5 minutes. The [React App](https://www.mathworks.com/help/thingspeak/react-app.html) is used for condition monitoring. You can monitor the data coming in from your devices and set up an alert when the data indicates something may need attention. For example, you could configure ThingSpeak to send an email when the humidity on your plant floor exceeds a certain value. More broadly, your analyses can trigger events that push data from ThingSpeak to other web applications like Salesforce via REST APIs.



* 1. My project



**REFERENCES.**

[1] https://thingspeak.com/pages/commercial\_learn\_more

[2] https://tinhte.vn/thread/review-blynk-ai-cung-lam-duoc-iot.3322306/

[3] https://www.semanticscholar.org/paper/Smart-Garden-Management-System-Shireen-Devi/dbb3b28fec4f415ec8899c8fb7715e8304a0944f