HO CHI MINH CITY, UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEER

Implementation and application of TDS sensor

Student 1: Phạm Minh Quang 1952410
Student 2: Trương Quang Thái 1952451

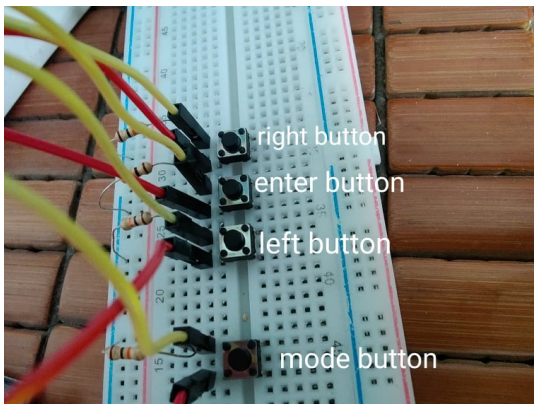
HỒ CHÍ MINH CITY, 01/01 / 2022

Content

1 Introduction

This project aims to measure the Total dissolved solids in water. And in order to do that , our team use ESP32 and TDS sensor . We also implemented some mode using buttons and displayed it on the LCD. For visualization purpose, we made an app to display the current value that had been published on the Thingspeak server by ESP32 devices

2 Gateway implementation



Above figure show the circuit with 4 buttons wich each button's function is commented in the picture. The 'mode button' controls 4 modes in this system:

The first is 'waiting mode': the system do nothing and wait user change to the next mode 'measuring'.

The second mode 'measuring mode': the system do measurement of TDS value throw TDS sensor and upload to thingspeak server in cyclical operation.

The third mode 'review data': the system get 10 most recent data which user can review it.

The fourth mode 'change measurement period': the system allow user to change the period of measurement in 'measuring mode'

2.1 Gateway configuration

2.1.1 Library use

Listing 1: Library use

```
1 import sys
2 import time
3 import network
4 import urequests
5 import machine
6 from i2c_lcd import I2cLcd
```

2.1.2 constant variable

Listing 2: static and dynamic variable

```
DURATION_FOR_AUTO_INCREASING = 3
BUFFER_DURATION = 0
DURATION_FOR_UPDATE = 15000
DURATION_FOR_10_MS = 10
update_time = time.ticks_ms()
update_10ms = time.ticks_ms()
index = 9
old_index = 8
# BUTTON buffer for debouncing
counterForBTN_Press30ms = {
"btn_mode": 0,
"btn_left": 0,
"btn_enter": 0,
"btn_right": 0
}

flagForBTN_Press30ms = {
"btn_mode": False,
"btn_left": False,
"btn_enter": False,
"btn_right": False
}
The counterForBTN_Press30ms and flagForBTN_Press30ms list use to ↩
    debounce the button when it's pressing.
```

2.1.3 network configuration

In this project we use the HTTP protocol to send and receive data throw 'GET' and 'POST' request

Listing 3: network configuration

```
sta = network.WLAN(network.STA_IF)
if not sta.isconnected():
  print('connecting to network...')
  sta.active(True)
  sta.connect('wifi id','wifi password')
  while not sta.isconnected():
    pass
print('network config:', sta.ifconfig())

```

```
10 HTTP_HEADERS = {'Content-Type': 'application/json'}
11 THINGSPEAK_WRITE_API_KEY = '9FCZF9HO7Z9H55N7'
12 response = ""
```

2.1.4 Pin configuration

Listing 4: Pin configuration

```
1  #define TDS Pin
2  TDS = machine.ADC(machine.Pin(34))
3  TDS.atten(machine.ADC.ATTN_11DB)
4  TDS.width(machine.ADC.WIDTH_10BIT)
5
6
7  #define LCD1602
8  DEFAULT_I2C_ADDR = 0x27
9  i2c = machine.I2C(scl=machine.Pin(22), sda=machine.Pin(21), freq↩
       =500000)
10 lcd = I2cLcd(i2c, DEFAULT_I2C_ADDR, 2, 16)
11
12
13 #define buttons
14 ModeButton = machine.Pin(14, machine.Pin.IN)
15 LeftButton = machine.Pin(16, machine.Pin.IN)
16 EnterButton = machine.Pin(17, machine.Pin.IN)
17 RightButton = machine.Pin(25, machine.Pin.IN)
18 # the 'mode button' use to switch between modes.
19 # the 'left button' and 'right button' use to switch displayed data ↩
       that received from server (10 datas).
20 # the 'enter button' use to confirm the period that user choose in '↩
       mode 4'.
```

2.2 helping function

2.2.1 switchMode function

The 'switchMode function' help to switch mode when the 'mode button' is pressed.

Listing 5: switchMode function

```
1 def switchMode(MODE):
2     global response
3     global update_10ms
4     global update_time
```

```
5       if MODE == 0:
6         MODE = 1
7         lcd.clear()
8         lcd.putstr('MEASURING....')
9       elif MODE == 1:
10      # when switch to mode 2 the variable 'response' recive the json ↩
            data from thingspeak server by an URL throw get request
11        response = urequests.get('https://api.thingspeak.com/channels↩
              /1558497/fields/1.json?api_key=1BLMFWISBOB60LD2&results=10')
12        lcd.clear()
13        lcd.putstr('RECEIVING DATA')
14        MODE = 2
15      elif MODE == 2:
16        lcd.clear()
17        lcd.putstr('   CHANGE MEASURING PERIOD')
18        MODE = 3
19      elif MODE == 3:
20        lcd.clear()
21        lcd.putstr('WAITING MODE')
22        MODE = 0
23      update_10ms = time.ticks_ms()
24      update_time  = time.ticks_ms()
25      return MODE
```

2.2.2 lcd display function

The 'lcd display function' help to display the informations to the LCD1602.

Listing 6: lcd display function

```
1   def lcd_display(TDS_value, mode):
2       lcd.clear()
3       if mode == 0:
4         lcd.putstr('waiting to turn on....')
5       elif mode == 1:
6         lcd.putstr('TDS:{} ppm'.format(TDS_value))
7       elif mode == 2:
8         lcd.putstr('old TDS:{} ppm'.format(TDS_value))
```

2.3 superloop for system progressing

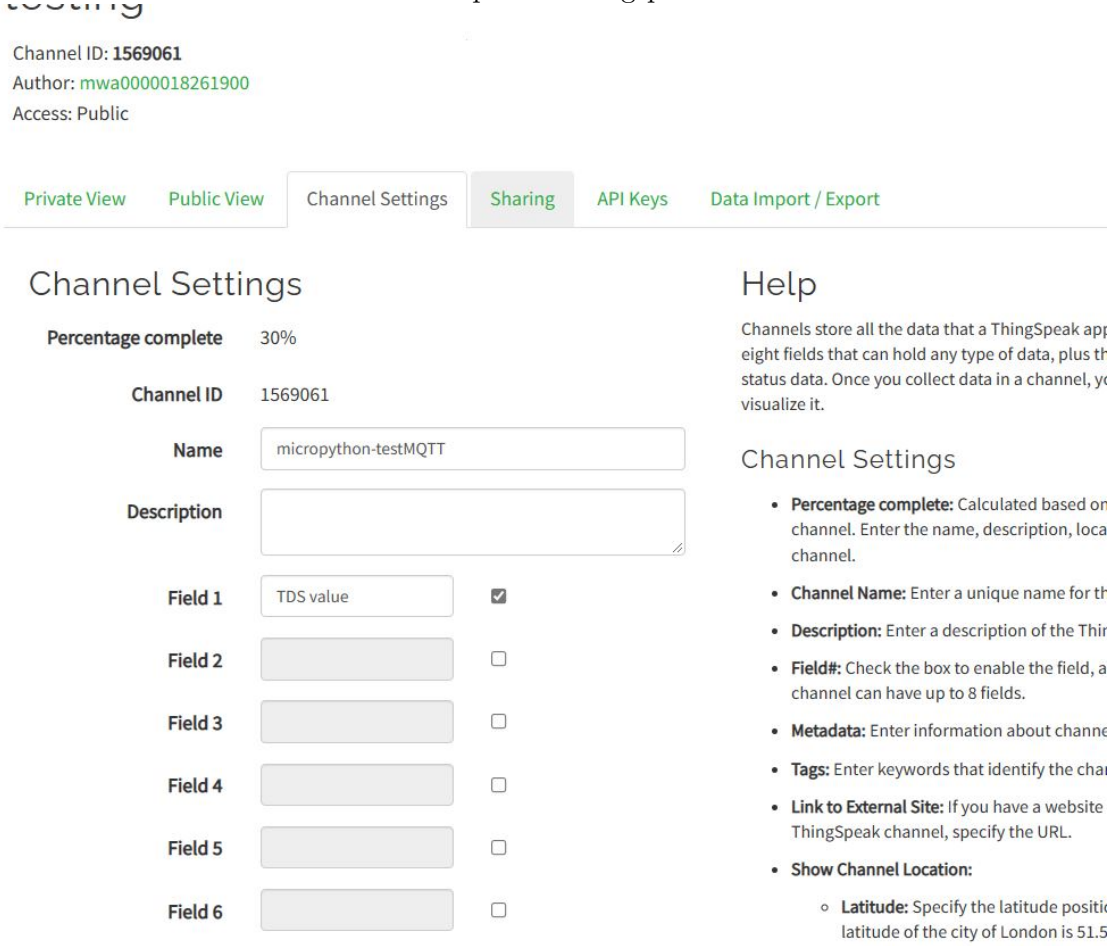
Listing 7: button checking

```
#the checking occure every 10ms
if time.ticks_ms() - update_10ms >= DURATION_FOR_10_MS:
    # a list use to read to value of button every 10ms
      btn_state={
       "btn_mode": 0,
       "btn_left": 0,
       "btn_enter": 0,
       "btn_right": 0
      }
      btn_state['btn_mode'] = int(ModeButton.value())
      btn_state['btn_left'] = int(LeftButton.value())
      btn_state['btn_enter'] = int(EnterButton.value())
      btn_state['btn_right'] = int(RightButton.value())
      # for loop to traverse all 4 buttons
      for btn in btn_state:
        if btn_state[btn] == 1 and flagForBTN_Press30ms[btn] == False:
        # button is pressed start to debounce
          if btn == "btn_mode":
            if counterForBTN_Press30ms[btn] < ↩
               DURATION_FOR_AUTO_INCREASING: #<30ms
            #debounce button for 30ms
              counterForBTN_Press30ms[btn] += 1
            else:
            #debounce complete, start progressing
              MODE = switchMode(MODE)
              if MODE == 2:
                index = 9
                old_index = index - 1
              elif MODE == 3:
                BUFFER_DURATION = int(DURATION_FOR_UPDATE /1000)
                lcd.clear()
                lcd.putstr('PERIOD:{} s'.format(BUFFER_DURATION))
              flagForBTN_Press30ms[btn] = True
          elif (btn == "btn_left" or btn == "btn_right") and (MODE == ↩
              2 or MODE == 3):
            if counterForBTN_Press30ms[btn] < ↩
               DURATION_FOR_AUTO_INCREASING:
              counterForBTN_Press30ms[btn] += 1
              #debounce button for 30ms
            else:
             #debounce complete, start progressing
              if btn == "btn_left":
                if MODE == 2 and index > 0:
                  index -= 1
                elif MODE == 3 and BUFFER_DURATION > 15:
```

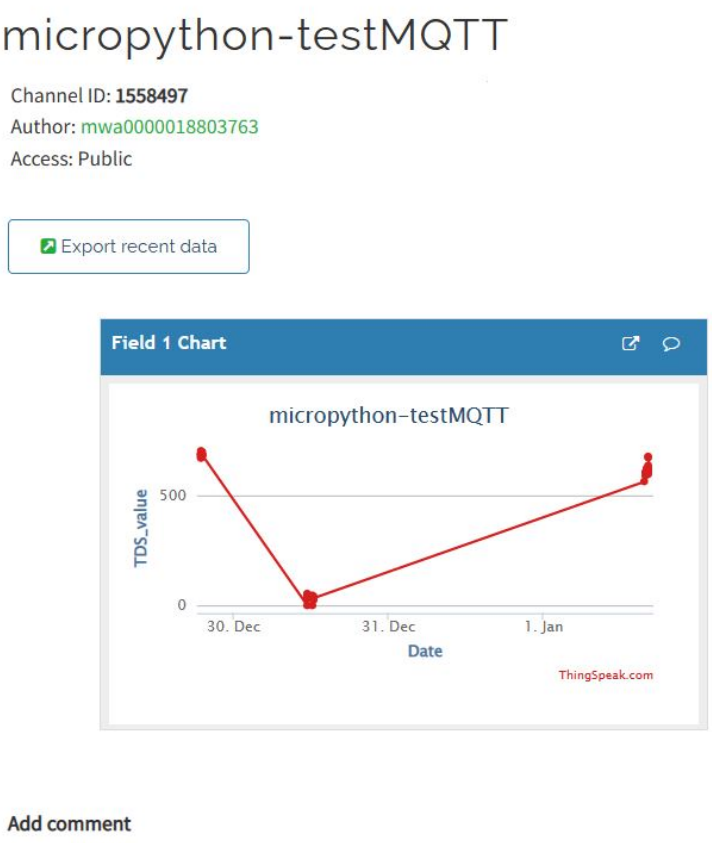
```
43                  BUFFER_DURATION -= 1
44                  lcd.clear()
45                  lcd.putstr('Period:{} s'.format(BUFFER_DURATION))
46                  print('PERIOD:{} s'.format(BUFFER_DURATION))
47              elif btn == "btn_right":
48                if MODE ==2 and index < 9:
49                  index+=1
50                elif MODE == 3:
51                  BUFFER_DURATION += 1
52                  lcd.clear()
53                  lcd.putstr('PERIOD:{} s'.format(BUFFER_DURATION))
54                  print('PERIOD:{} s'.format(BUFFER_DURATION))
55              flagForBTN_Press30ms[btn] = True
56          elif btn == "btn_enter" and MODE == 3:
57            if counterForBTN_Press30ms[btn] < ↩
                 DURATION_FOR_AUTO_INCREASING:
58              counterForBTN_Press30ms[btn] += 1
59            else:
60              DURATION_FOR_UPDATE = int(BUFFER_DURATION * 1000)
61              lcd.clear()
62              lcd.putstr('PERIOD IS SET')
63              flagForBTN_Press30ms[btn] = True
64              print('NEW PERIOD IS:{} s'.format(BUFFER_DURATION))
65        else:
66          if btn_state[btn] == 0:
67            counterForBTN_Press30ms[btn] = 0
68            flagForBTN_Press30ms[btn] = False
69      update_10ms = time.ticks_ms()
```

3 Server configuration

First we need to create a field in the topic on Thingspeak cloud server



After published the data onto the server via ESP32 , this is the result we will get:

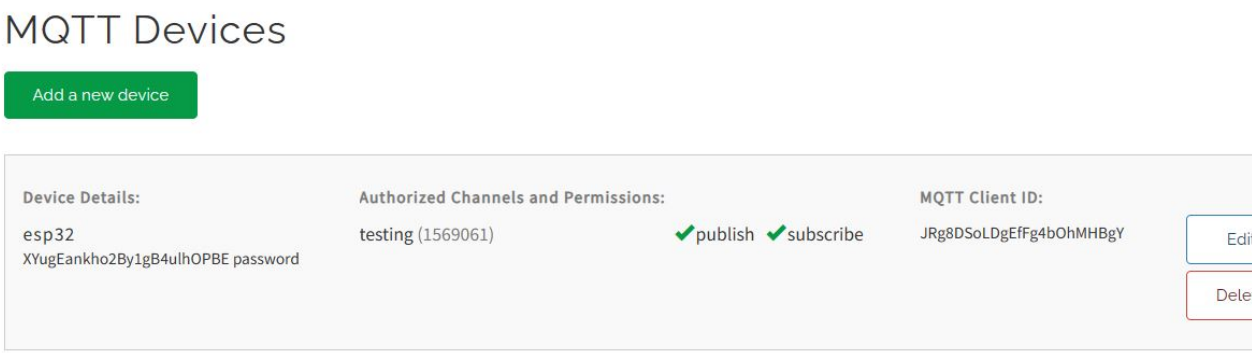


4 Monitoring application

In order to get the current TDS value on the server, our team use MQTT to received the value and display it on the app.

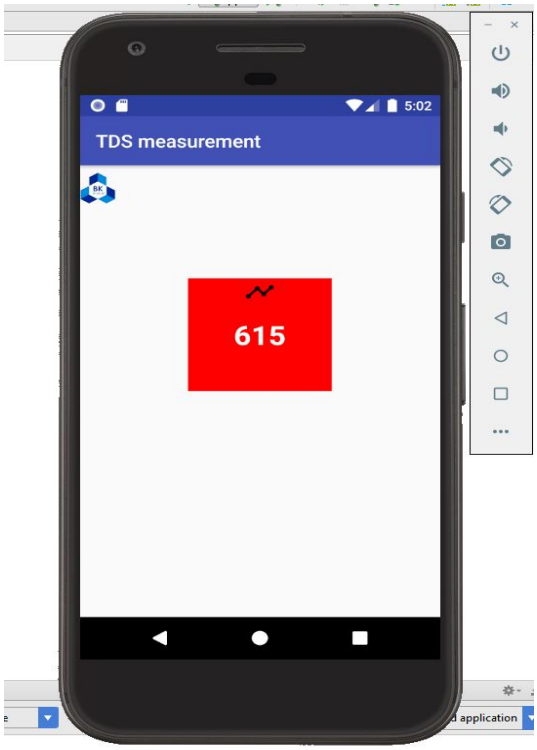
4.1 MQTT Thingspeak configuration

Thingspeak server support MQTT protocol through a seperated section and needed to be config-



ured

4.2 The app



5 Demo youtube link

Click to enter Demo link

Tài liệu

[Web] Arduino, https://www.arduino.cc/