به نام خدا

پروژه: سطل زباله هوشمند

استاد:دکتر امیرمهدی منزه

اعضا گروه:

پوریا رحیمی

فاطمه عسکری

شرح پروژه:

هدف از انجام این پروژه این هست که به فرایند تفکیک زباله کمک کنیم در واقع این سطل زباله با توجه با اینکه یک سری زباله خاص را قبول میکند به ما کمک می کند که وقت و هزینه کمتری برای تفکیک زباله و بازیافت مصرف کنیم.

یک ماژول بارکد اسکنر داریم که بارکد کالاها را اسکن می کند سپس از طریق ESP32 به WIFI متصل میشویم و از طریق بروکر mqtt در دیتابیس بارکد کالا را سرچ میکند اگر بارکد موجود بود در سطل زباله به مدت ۲۰ ثانیه باز میشود و می تواند زباله را درون سطل زباله بیاندازد و اگر بارکد در دیتابیس موجود نبود در سطل زباله باز نمی شود و یک LED روشن میشود.

نحوه پیاده سازی:

در ابتدا با استفاده از کد پایتون و sqllite دیتابیس خود را ایجاد می کنیم و چند نمونه نیز برای تست در آن insert میکنیم:

```python
import sqlite3

conn = sqlite3.connect('mqttrecycle.db')
cursor = conn.cursor()

cursor.execute("DROP TABLE IF EXISTS barcodedata")

cursor.execute('''CREATE TABLE IF NOT EXISTS barcodedata
                  (id INTEGER PRIMARY KEY, barcode TEXT, type INTEGER)''')

data = [
    (1, '6930358609225', 0),
    (2, '8690146123637', 0),
    (3, '6260176700829', 1),
    (4, '8901057335645', 1),
    (5, '6001567179851', 0),
    (6, '6260111310106',0)
]

for record in data:
    conn = sqlite3.connect('mqttrecycle.db')
    cursor = conn.cursor()
    cursor.execute('INSERT INTO barcodedata VALUES (?, ?, ?)', record)
    conn.commit()
    conn.close()
```

در مرحله بعد یک برنامه را با استفاده از پایتون پیاده‌سازی می‌کنیم که با استفاده از پروتکلMQTT ، به یک بروکر MQTT متصل می‌شود و برای دریافت بارکدها از یک دستگاه ESP32 استفاده می‌کند. سپس بارکد دریافت شده را در پایگاه داده SQLite سرچ می‌کند و نوع متناظر با هر بارکد را در یک تاپیک دیگر در بروکر MQTT منتشر می‌کند.

در ابتدا، متغیرهای مربوط به بروکر MQTT و اطلاعات اتصال به آن (آدرس بروکر، پورت، نام کاربری و رمز عبور) تعریف می کنیم:

```
mqtt_broker = 'broker.emqx.io'
mqtt_port = 1883
mqtt_topic = 'esp32/barcode'
mqtt_topictype='esp32/type'
mqtt_user = 'emqx'
mqtt_password = 'public'

database = 'mqttrecycle.db'
table = 'barcodedata'
```

سپس، یک اتصال به پایگاه داده SQLite برقرار می‌شود و از طریق اجرای دستورات SQL روی جدول مربوطه، بررسی و استخراج نوع متناظر با بارکد انجام می‌شود.

تابع check_barcodeورودی بارکد را دریافت می‌کند و ابتدا بارکد را تمیز می‌کند (حذف فاصله‌های اضافی) و سپس با استفاده از یک دستورSQL ، نوع متناظر با بارکد را در پایگاه داده جستجو می‌کند. اگر بارکد در پایگاه داده وجود داشته باشد، نوع متناظر با آن بارکد را از پایگاه داده خوانده و در یک موضوع MQTT منتشر می‌کند. در غیر این صورت، مقدار ۲ را در موضوع MQTT منتشر می‌کند.

```
def check_barcode(barcode):
    barcode = barcode.strip()
    if (barcode=='0'):
        print("scdv")
    print(barcode)
    cursor.execute('SELECT type FROM {} WHERE barcode = ?'.format(table), (barcode,))
    result = cursor.fetchone()

    if result is not None:
        type_value = result[0]
        mqtt_client.publish(mqtt_topictype, str(type_value))
    else:
        mqtt_client.publish(mqtt_topictype, '2')
```

توابع on_connectو on_messageنیز به ترتیب برای رسیدن به اتصال با بروکر MQTT و دریافت پیام‌ها از آن استفاده می‌شوند. در تابع on_connect، عملیات اشتراک‌گذاری برای دریافت پیام‌های مربوط به بارکدها انجام می‌شود و تابع on_messageهنگام دریافت یک پیام جدید، بارکد را از آن استخراج کرده و به تابع check_barcodeارسال می‌کند.

در نهایت، یک نمونه از کلاس mqtt.Clientایجاد می‌شود و توابع on_connectو on_messageبه آن اختصاص داده می‌شوند. سپس با استفاده از متدهای connectو loop_forever، اتصال به بروکر MQTT برقرار می‌شود و برنامه در یک لوپ بی نایت منتظر دریافت پیام‌ها می‌ماند.

در واقع هر وقت که بارکد یک کالا توسط بارکد اسکنر خوانده شد esp32 در یک تاپیک بارکد کالا را publish می کند و در این کد پایتون آن را subscribe می کند و با توجه به اینکه این بارکد در دیتابیس هست یا نه پیامی را در یک تاپیک دیگر publish میکند و esp32 ان را subscribe میکند که عملیات مورد نظر را انجام دهد.

کدی که برای eesp32 در Arduino زدیم نیز به صورت زیر است:

این کد با استفاده از پروتکل MQTT با یک بروکر عمومی به نام EMQX ارتباط برقرار می‌کند. برنامه شامل اتصال به شبکهWi-Fi ، ارسال و دریافت پیام‌ها از بروکرMQTT ، کنترل یک سروموتور برای تغییر جهت و چرخش، و کنترل یک LED است.

اجزای اصلی کد عبارتند از:

- تنظیمات شبکه Wi-Fi: شامل نام شبکه (SSID) و رمز عبور (password) است.
- تنظیمات بروکر MQTT: شامل آدرس(mqtt_broker) ، پورت(mqtt_port) ، نام کاربری (mqtt_username)و رمز عبور (mqtt_password) بروکر MQTT است.

```
// WiFi
const char* ssid = "Pooria";
const char* password = "Mrx00120012";

// MQTT Broker
const char *mqtt_broker = "broker.emqx.io";
const char *topic = "esp32/barcode";
const char *topictype = "esp32/type";
const char *mqtt_username = "emqx";
const char *mqtt_password = "public";
const int mqtt_port = 1883;

WiFiClient espClient;
PubSubClient client(espClient);
```

- متغیرهای سروموتور: شامل myservo (برای کنترل سروموتور) و pos (برای زاویه فعلی سروموتور) است.

```
ESP32PWM::allocateTimer(0);
ESP32PWM::allocateTimer(1);
ESP32PWM::allocateTimer(2);
ESP32PWM::allocateTimer(3);
myservo.setPeriodHertz(50);// Standard 50hz servo
myservo.attach(18, 500, 2400);   // attaches the servo on pin 18 to the servo object
                                 // using SG90 servo min/max of 500us and 2400us
                                 // for MG995 large servo, use 1000us and 2000us,
                                 // which are the defaults, so this line could be
```

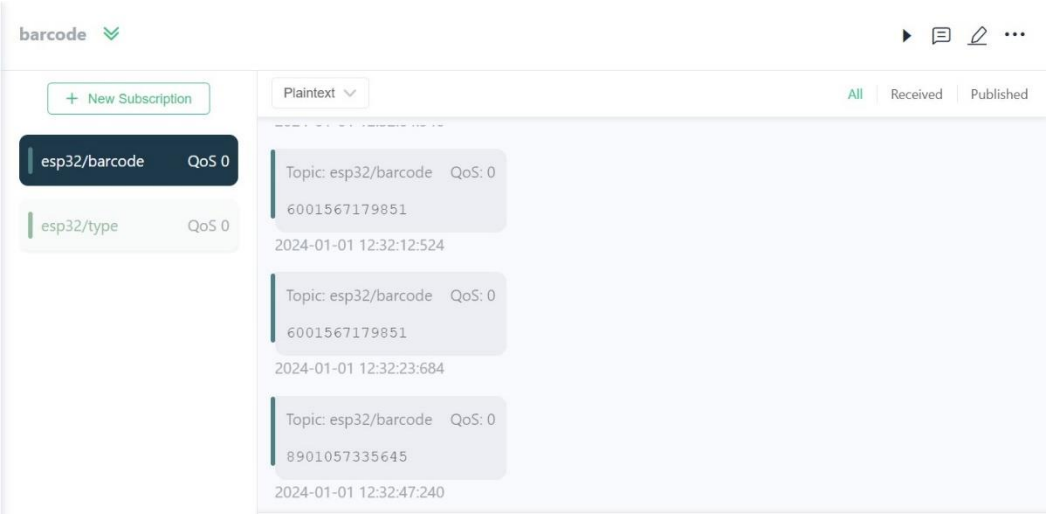
- تابع setup: شامل تنظیمات اولیه برنامه از جمله اتصال به شبکه Wi-Fi و بروکرMQTT ، تنظیمات سروموتور و تنظیمات دیگر است.
- تابع callback :برای پردازش پیام‌های دریافتی از بروکر MQTT استفاده می‌شود. این تابع به طور خاص برای کنترل سروموتور و LED فراخوانی می‌شود.
- تابع turner: برای چرخش سروموتور به زوایای مختلف استفاده می‌شود.

```
void turner(){
    for (pos = 0; pos <= 45; pos += 1) {
        myservo.write(pos);
        delay(15);
      }
      delay(20000);
      for (pos = 45; pos >= 0; pos -= 1) {
        myservo.write(pos);
        delay(15);
      }
}
```

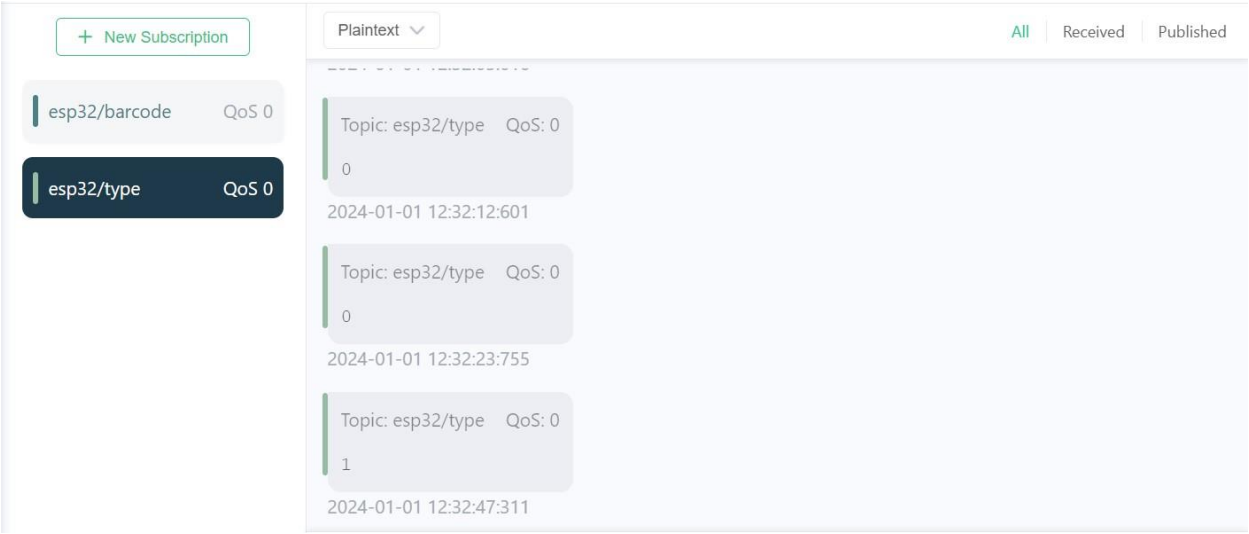
- تابع loop :برای اجرای حلقه اصلی برنامه استفاده می‌شود. در این حلقه، پیام‌های ارسالی از سنسور بارکد خوان خوانده شده و به بروکر MQTT ارسال می‌شوند. همچنین، پیام‌های دریافتی از بروکر MQTTپردازش می‌شوند.

هم چنین برای دیدن پیام هایی که publish و subscribe میشوند از برنامه MQTTX استفاده کردیم:

وقتی بارکد را می خوانیم و ان را در تاپیک esp32/barcode پابلیش می کنیم پیام ها به صورت زیر است:



و هم چنین موقعی که type کالا را در تاپیک esp32/type پابلیش می کنیم پیام ها به صورت زیر است:



سخت افزار پروژه ما نیز به صورت زیر است: