به نام خدا

گزارش فاز دوم پروژه

سیستم های نهفته و بی درنگ

موضوع پروژه: دزدگیر ماشین

پریسا علائی، 97522175

غزل زمانی نژاد، 97522166

در این پروژه با پردازنده 32esp و ماژول های دیگر که جلوتر معرفی میشود سعی به ساختن یک دزدگیر ماشین پرداخته ایم.

مراحل پیش بردن پروژه:

- ابتدا 32esp را سعی کردیم به لپ تاپ وصل کنیم که نیاز بود که 2102device cp را به صورت جداگانه در os خود نصب کنیم.
- وصل کردن 6050mpu: ابتدا نیاز بود که این ماژول را لحیم کنیم تا بدون قطع و وصل شدن بتواند به کار خود ادامه دهد. این ماژول 3 رنجAccelerometer وGyro و FilterBandwidth دارد که هر کدام به ترتیب عددی را میگیرند و با ست کردن آن میتوان میزان حساسیت را در این ماژول تنظیم کرد. در اینجا نیاز داشتیم که شدت میزان لغزش را تنظیم کنیم پس نیاز به استفاده از accelerometer بود و آن دو رنج دیگر برای ما مهم نبود. این شتاب سه عدد در راستای ارتفاع و عرض و طول میداد و برای آن که ما به عددی واحد برسیم هر کدام را به توان دو رساندیم و جمع کردیم و سپس جذر گرفتیم و به یک شتاب واحد رسیدیم. در اینجا هر چه رنج انتخابی کوچک تر باشد شدت حساسیت این ماژول بیشتر خواهد بود و ما با درنظر گرفتن رنج 2 به عدد واحد 10.28 برای حالتی که تکانی نداریم رسیدیم.
- راه اندازی buzzer: در این قسمت نیاز داشتیم که هرموقع تکان بیشتری نسبت به حالت عادی داشتیم بازر را به صدا در آوریم پس ما در این حالت روی 10.5 یک آستانه تعریف کردیم که اگر از آن بیشتر شد صدای بازر بلند شود.
- راه اندازی sim800l: این ماژول که سیم کارت میگیرد و اجازه ی اس ام اس دادن به ما و استفاده از هات اسپات برای وصل شدن 32esp به اینترنت استفاده کردیم.
    - اس ام اس زدن: در اینجا اگر ماشین خاموش باشد و الارم ماشین کار کند برای ما یک بار اس ام اس زده میشود که الارم ماشین درحال کار کردن است . و وقتی بازر شروع به الارم زدن میکند به فرد یک اس ام اس فرستاده میشود که بازر در حال کار کردن است (این یعنی وقتی ماشین خاموش است یکی به زور میخواهد در را باز کند)

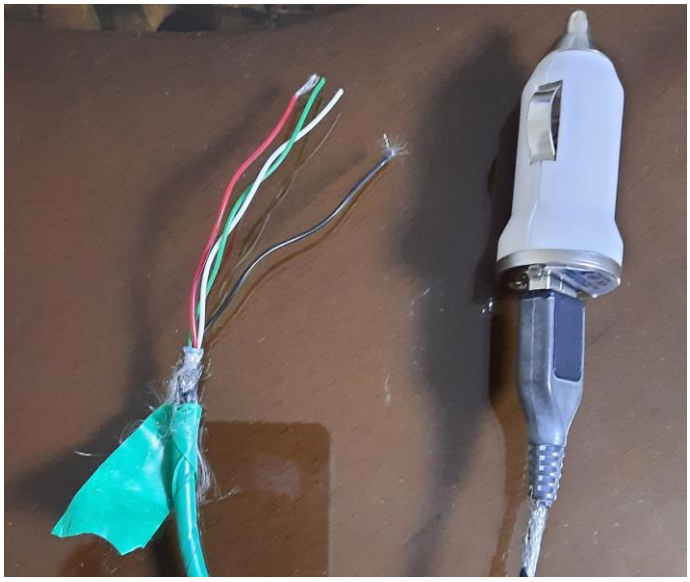
      برای ارسال اس ام اس از یک سری at command استفاده کردیم که شماره سیم کارتی که میخواهیم اس ام اس کند را میگفت و دستورات لازم برای آماده شدن ماژول برای ارسال اس ام اس و متصل بودن سیم کارت به اینترنت را وارد کردیم .

- وصل شدن به اینترنت و هات اسپات کردن : در این پروژه چون ماشین در مکان هایی قرار میگیرد که ممکن است وای فای در اختیار نداشته باشد و ما برای وصل شدنبه سرور نیاز داریم که همواره به اینترنت وصل یاشیم به این ماژول نیاز داریم که با استفاده سیم کارت همواره به ما اینترنت را بدهد. برای راه اندازی هات اسپات از tinyGSM استفاده کردیم که باید برای وصل شدن 32esp و sim800l پین های rx و tx را تعریف میکردیم. و یک مودم که sim800l بود را نیز تعریف کردیم تا بتوانیم از اینترنت سیم کارت استفاده کنیم.
- در اینجا برای ارسال دستورات at از دو سریال متفاوت 32esp استفاده کردیم . در کل 32esp دارای 3 سریال است که دو تای آن خالی است و ما از آن دو تا استفاده کرده ایم و دستوراتی که در هر سریال زده میشود را در آن یکی تبادل کرده ایم و هردو سریال را آپدیت نگه میداریم.

- رله: برای خاموش و روشن کردن ماشین از رله استفاده میکنیم که مثل یک سوئیچ است. یک طرف آن به کنترل کننده که 32esp است وصل میشود و طرف دیگر به ولتاژی که قرار است کنترل شود. در اینجا ما یک اشتباه کردیم و رله ای که ولتاژ ورودی آن برای کنترل کننده 12 ولت است خریدیم که متاسفانه با این رله نتوانستیم کار کنیم.
نکته ای دیگر این است که قرار شد که به جای خاموش و روشن کردن ماشین، یک led را خاموش و روشن کنیم.
در اینجا کدی که برای این قسمت زده شده است این گونه است که ما از سرور با کمک Adafruit_MQTT_Subscribe پیامی که از طرف سرور آمده است را دریافت میکنیم و دستورات خاموش و روشن بودن led را دریافت و با توجه به آن میگوییم که کی رله مثلا سوییچ به normally close و کی سوییچ به normally open کند.

- وصل کردن به ماشین: برای بررسی روشن / خاموش بودن ماشین، از فندکی ماشین ولتاژ را بررسی می کنیم. در صورتی که ماشین روشن باشد، عدد پین مربوطه 4095 و در غیر این صورت 0 خواهد بود. برای این کار، از یک تبدیل فندکی به usb با ولتاژ خروجی 5 ولت استفاده کردیم. سپس به جهت اینکه ممکن است مدار با ولتاژ ورودی 5 ولت بسوزد، با تقسیم مقاومتی آن را به 3 ولت کاهش دادیم. یک سر مقاومت را به led و سر دیگر آن را به پین 32D که یک پین ADC است وصل کردیم. از

یک کابل شارژ استفاده کردیم و سیم یک سر آن را لخت کردیم. از دو سیم قرمز و مشکی آن برای اتصال به مدار بهره گرفتیم. سیم قرمز را به یک سر led و سیم مشکی را به مقاومت وصل کردیم. سپس مقدار ولتاژ دریافتی را در car_wire_value ذخیره می کنیم.



```
car_wire_value = analogRead(car_wire);
```

- استفاده از wifi
- برقرار ارتباط با سرور MQTT: برای ارسال و دریافت اطلاعات به سرور، از کتابخانه Adafruit_MQTT استفاده شد. در ابتدا یک client برای برقرار ارتباط ایجاد کردیم و به آن اطلاعات سرور، نام کاربری و پسورد را دادیم.

```
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER,
AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);
```

سپس اطلاعات ارسالی به سرور را به عنوان publish و اطلاعات دریافتی از سرور را به عنوان subscribe، initialize کردیم.

```
Adafruit_MQTT_Publish car_state = Adafruit_MQTT_Publish(&mqtt,
AIO_USERNAME "/car_state");
Adafruit_MQTT_Publish alarm_state =
Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/alarm_state");
```

```
Adafruit_MQTT_Publish buzzer_state =
Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/buzzer_state");
// get on/off from server
Adafruit_MQTT_Subscribe onoffbutton =
Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME
"/ONOFF_FEED");
```

در تابع check_car_signal، بعد از بررسی ولتاژ دریافتی از فندکی ماشین وضعیت ماشین و دزدگیر را به همراه ساعتی که از سرور ntp دریافت کرده ایم به سرور ارسال می کنیم.

همچنین در تابع buzzer_sound، وضعیت بازر را به سرور گزارش می دهیم.

در تابع receiveCommand، در یک حلقه از سرور می خوانیم و در صورتی که topic آن صحیح باشد، مقدار را به عنوان خروجی تابع باز می گردانیم. در غیر این صورت مقدار -1 را برمی گردانیم.

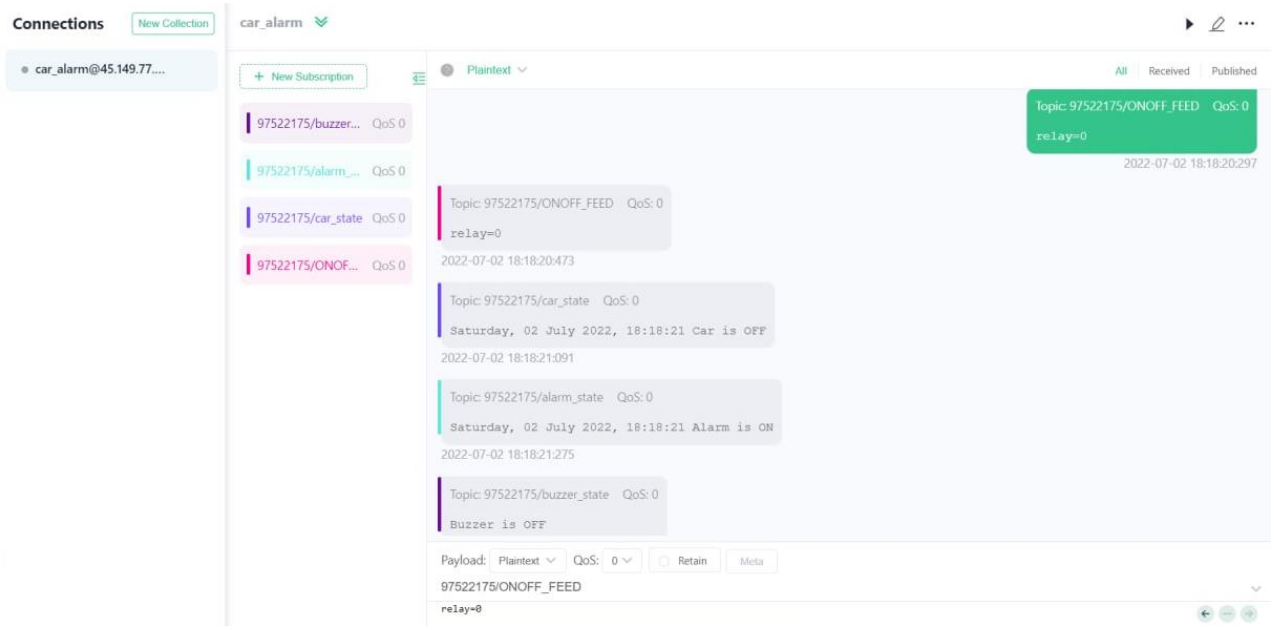
```
int receiveCommand() {
  int clientSt = -1;
  // this is our 'wait for incoming subscription packets' busy subloop
  Adafruit_MQTT_Subscribe *subscription;
  while ((subscription = mqtt.readSubscription(10))) {
    if (subscription == &onoffbutton) {
      Serial.print(F("Got: "));
      Serial.println((char*)onoffbutton.lastread);

      char relay_val[7];
      strcpy(relay_val, (char*)onoffbutton.lastread);
      Serial.print("receiveCommand: ");
      Serial.println(relay_val[6]);
      return (int)relay_val[6];
    }
  }
  return clientSt;
}
```

در تابع loop به صورت مداوم تابع receiveCommand را صدا می زنیم و در صورتی که مقدار آن غیر از -1 باشد، مقدار آن را بررسی می کنیم. اگر 1 باشد آن را به عنوان وصل کردن رله و در نتیجه روشن شدن led در نظر می گیریم. اگر 0 باشد آن را به عنوان قطع کردن رله و در نتیجه خاموش شدن led در نظر می گیریم.

```
//get relay_value from server
  int val = receiveCommand();
  if(val != -1){
    Serial.print("server relay value");
    Serial.println(val);
    if(val-48 == 0){
      digitalWrite(relay, LOW);
    }
    if(val-48 == 1){
      digitalWrite(relay, HIGH);
    }
  }
```

در زیر تصویری از نرم افزار MQTTX که برای مانیتور کردن سرور به کار رفت دیده می شود:



گزارش کاملتر در فاز سوم پروژه ارائه خواهد شد.

منابع

- ESP32 MPU-6050 Accelerometer and Gyroscope (Arduino) | Random Nerd Tutorials
- Active and Passive Buzzer for Arduino, ESP8266 and ESP32
- ESP32 SIM800L: Publish Data to Cloud without Wi-Fi | Random Nerd Tutorials
- سورس برنامه + GSM Sim800L بخش اول تست شبکه Sim800L آموزش راه اندازی ماژول
- ESP32 Hardware Serial2 Example | Circuits4you.com
- سیم کارت - دیجی اسپارک GPRS بخش پنجم: اینترنت SIM800L راه اندازی و کار با ماژول
- ESP32 SIM800L GSM Module Tutorial - How SIM800L Module Works and Interfacing it with ESP32
- Intro to Adafruit MQTT | MQTT, Adafruit IO & You! | Adafruit Learning System
- GSM Modem - سیسوگ - Sisoog دریافت و تنظیم زمان دقیق توسط