組み込み実験

5EC 中野将生

2019/6/26

1　担当部分の詳細回路図と担当部分のソースコード及び処理の説明

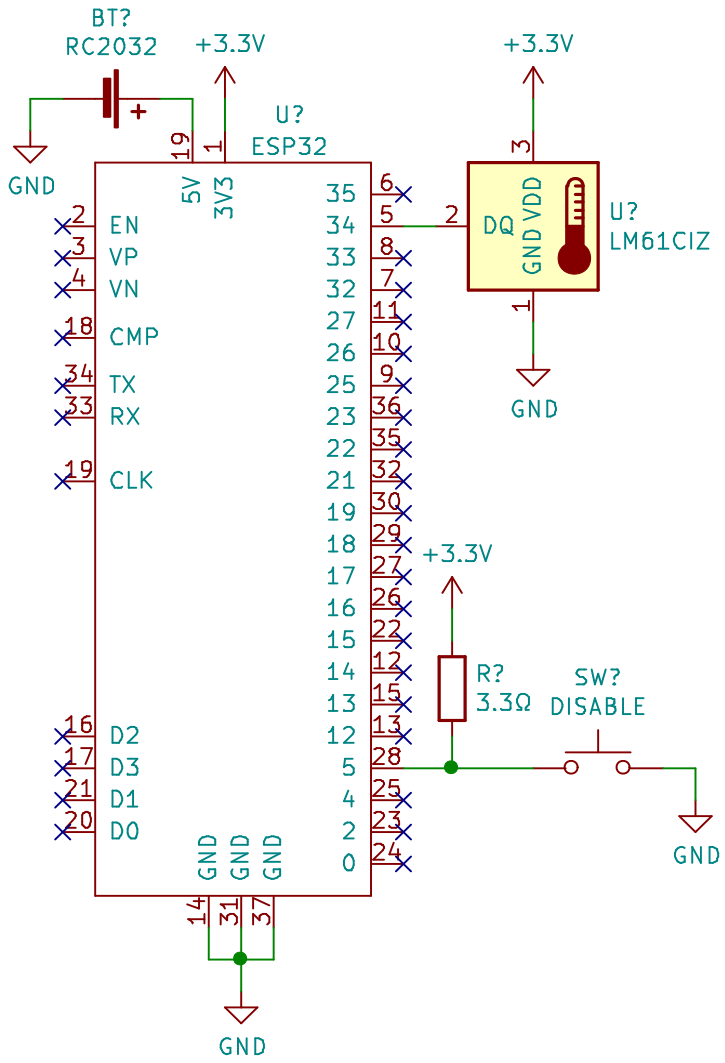

図 1：詳細回路図

```
#include <WiFi.h>
#include <HTTPClient.h>

const char* SSID = "TestAP";
const char* PASSWORD = "";
const char* ENDPOINT = "https://us-central1-frid-vue.cloudfunction-
```

```
s.net/updateTemp";
const int INTERVAL = 5 * 1000;
const int ADC_INPUT = 34;


const char* DEVICE_ID = "0";

/* Required: arduino-esp32
 * pinout:
 *   ESP32 GPIO4 -             LM61ciz center
 *   ESP32 3.3V  - IM920c Vcc, LM61ciz Vcc
 *   ESP32 GND   - IM920c GND, LM61ciz GND
 */

void setup() {
  WiFi.disconnect(true);
  delay(1000);
  Serial.begin(115200);
  Serial.print("Attempting to connect");
  while(WiFi.status() != WL_CONNECTED) {
    WiFi.begin(SSID, PASSWORD);
    Serial.print(".");
    delay(500);
  }
  Serial.println("connected: ");
  Serial.print(WiFi.localIP());
  pinMode(INPUT, ADC_INPUT);
}

String buildBody(int id, float temp) {
  auto json = String("{\"id\":\"");
  json.concat(String(id));
  json.concat(String("\", \"temperature\":"));
  json.concat(String(temp));
```

```
  json.concat(String("}"));
  return json;
}

void loop() {
  Serial.println(analogRead(ADC_INPUT));
  float voltage = analogRead(ADC_INPUT) * 3.3f / 4096.0f;
  float temperature = (voltage - 0.6) * 100.0f;

  HTTPClient http;
  http.begin(ENDPOINT);
  http.addHeader("Content-Type", "application/json");
  auto body = buildBody(0, temperature);
  auto httpResponseCode = http.POST(body);
  http.end();
  /*
   * Try this method in below issue, if Arduino failed to sene POST
header.
   * https://github.com/espressif/arduino-esp32/issues/2670#issuecommen-
t-483289542
   */
  Serial.print("temp: ");
  Serial.println(voltage);
  Serial.println(temperature);
  Serial.print("http status code: ");
  if (httpResponseCode < 0) {
    Serial.println(http.errorToString(httpResponseCode));
  }
  else {
    Serial.println(httpResponseCode);
  }
  for (int i = 0; i * 10 < INTERVAL; ++i) {
    if (digitalRead(5) == LOW) {
      http.begin("https://us-central1-frid-vue.cloudfunctions.net/
```

```
disableFood");
      Serial.println("disable");
      http.addHeader("Content-Type", "application/json");
      auto json = String("{\"id\":\"0\"}");
      http.POST(json);
      http.end();
    }
    delay(10);
  }
}
```

図 2：本体側ソースコード

```
import * as functions from 'firebase-functions';
import * as admin from 'firebase-admin';
admin.initializeApp();

// // Start writing Firebase Functions
// // https://firebase.google.com/docs/functions/typescript
//
// export const helloWorld = functions.https.onRequest((request, response)
=> {
//  response.send("Hello from Firebase!");
// });

const firestore = admin.firestore();

export const updateTemp = functions.https.onRequest(async (request,
response) => {
  console.log(request.body);
  await firestore.collection("foods").doc(request.body.id).update({
    temperature: request.body.temperature
  });
  response.sendStatus(200);
```

```
});

export const disableFood = functions.https.onRequest(async (request,
response) => {
  await firestore.collection("foods").doc(request.body.id).update({
    is_active: false
  });
  response.sendStatus(200);
});
```

図 3 :Cloudfunctions 側ソースコード

まず本体側の回路図を図1に示す。34 番ピンの ADC で温度計測 IC の出力を読み、5 番ピンで登録解除ボタンを監視している。 図2には Arduino で書かれた ESP32、 本体側のソースコードを示している。 基本的に WiFi に接続し、HTTPClient を使って POST でクエリを cloudfunctions に投げているだけだが、温度更新は 5 分に一度、登録解除は 10 ミリ秒毎に監視して登録解除の場合のみ API を叩くように設定している。Cloudfunctions 側のソースコードは図3である。Firebase の admin API でデータベースを処理している。あくまでプロトタイピングなので認証は行っていない。JavaScript ではなく TypeScript で書いている点以外には特筆することはない。

2　この製作おいて自分の担当した部分と自分の果たした役割

実際に手を動かした箇所としては、

- 本体側回路設計
- 本体側回路実装
- 本体側ファームウェア作成
- サーバー側の本体向け API 作成

の四点が挙げられる。 また、 チームリーダーとしてスケジュールに合わせて目標の再設定を行った。 一チームメンバーとしては、使用する技術、フレームワークを選定し、実際に Firebase を使用した。 フロントエンドは当初 Elm を使う予定だったが渡辺の判断で Vue.js に切り替えた。

3　自身のサクセスレベルに対する評価

前半は編入試験で忙しく後半は何かと用事が立て込むこともあったが、プロダクトに対する情熱がやや失われて画像認識系へと手を付けられなかったのは残念に思う。Level2 は達成したので及第点ではあると思っている。

4　考察等を含めた感想

4.1　モチベーションなどについて

ある程度チーム全体で目標を共有しやすいこと、授業の目的に沿うことを意識してアイデアを練ったが、自分にとって興味があることを少し犠牲にしてしまった結果後半ではプロダクトへの情熱が失われてしまった。ただ一時的であっても情熱を失っても進捗を進める力は、今後企業での開発、大学、大学院での研究で求められていく物であると思えるので、その点を鍛えられたのは良かったと思う。

4.2　IoT について

今回初めてマイコンでインターネットを扱ったが、思いの外簡単なコードで基本的な REST API の操作が出来ることに驚いた。一方で図2中の GitHub の issue を貼ったコメントにあるように、まだ十分に動作せず内部のソースコードとは言わずとも GitHub で issue を検索する事が求められるなど、メジャーなユーザランドで動作する環境では中々置きないような事が発生し中々上手くいかなかった。