به نام خدا

دانشکده مهندسی برق و کامپیوتر

سامانه تعبیه شده هوشمند پایش و تحلیل سیگنال‌های الکتروکاردیوگرافی و اعلام هشدار احتمال بروز خطر

پوریا پوریگانه، علیرضا صالحی حسین آبادی

علیرضا کشتی آرا، مهدی شیروانی

بهمن ماه ۱۴۰۰

فهرست مطالب

فهرست اشکال

بخش ۱:
کلیات پروژه

۱-۱: مقدمه

پایش و بررسی وضعیت بیماران از جمله مواردی است که به صورت مداوم و با فاصله زمانی مشخص توسط پزشک و پرستار در مراکز درمانی انجام می‌شود و با توجه به وضعیت بیمار نسبت به پیشگیری از حوادث بعدی و یا درمان موارد رخ داده شده اقدام می‌شود.

در این پایش وضعیت، عمدتاً بیشینه و کمینه فشار خون، تعداد ضربان قلب در هر دقیقه، درصد اکسیژن محلول در خون، الگوی انقباض ماهیچه‌ی قلب، و الگوی تنفس بیمار توسط کادر درمانی نظارت می‌شود. در میان موارد بیان‌شده الگوی انقباض ماهیچه‌ی قلب در مواقع بحران با سرعت بیشتری نسبت به سایر پارامترها تغییر می‌کند در عین حال که تحلیل این الگوها نیازمند افراد با تجربه و خبره است. با توجه به موارد بیان شده، طراحی سامانه‌ای که به صورت برخط این الگو را بررسی و تحلیل نماید و در مواقع احتمال بروز خطر به کادر درمان هشدار دهد دارای اهمیت قابل توجهی خواهد بود به خصوص که احتمال خطاهای انسانی را نیز کاهش خواهد داد.

افزایش توانایی‌های سیستم‌های تعبیه شده در عین قیمت نسبتا مناسب آن‌ها و نیز ابعاد و اندازه جمع و جور و انعطاف‌پذیری آن‌ها، ذهن را به سمت استفاده از این سیستم‌ها در کاربردهای مشابه موارد بالا می‌کشاند.

برای طراحی و پیاده‌سازی سامانه‌ی معرفی‌شده نیاز به تجهیزاتی هست که داده‌های الگوی انقباض قلب را به صورتی قابل فهم و استفاده برای یک سیستم تعبیه شده دریافت نماید، که خوشبختانه این تجهیز با نام حسگر الکتروکاردیوگرافی سالیان متمادی در حال استفاده است.

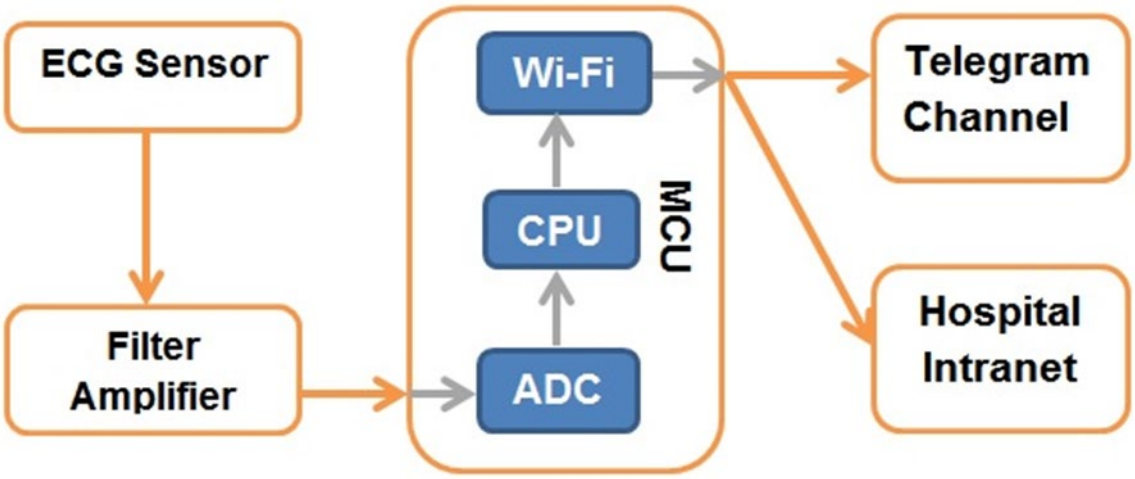
ماهیچه‌ی قلب هنگام انقباض، سیگنال‌هایی الکتریکی با دامنه ۰/۴ – ۳ mV و فرکانس ۰/۵ – ۱۰۰Hz تولید می‌کند که این سیگنال‌ها با استفاده از حسگر ECG قابل دریافت و بررسی هستند. همانطور که بیان شد این سیگنال‌ها از نوع ولتاژ آنالوگ و با دامنه نسبتا پایین هستند که برای پردازش و بررسی آن‌ها نیاز به تبدیل آن‌ها به مقادیر دیجیتال داریم.

پس از دریافت سیگنال‌ها و تبدیل آن‌ها به مقادیر دیجیتال، اکنون الگوهای انقباضی به گونه‌ای قابل فهم و تفسیر برای سیستم‌های تعبیه‌شده و کامپیوتری هستند. در ابتدا اشاره شد که تحلیل و بررسی الگوهای انقباضی ماهیچه قلب، فرایندی تجربی و وابسته به نظر شخصی فرد خبره است، لذا امکان انجام آن برای یک سیستم دیجیتال با روش‌های سنتی قابل انجام نیست. خوشبختانه افزایش توان پردازشی سخت‌افزارها و نیز رواج استفاده از هوش مصنوعی و سیستم‌های مبتنی بر یادگیری و پیشرفت آن‌ها، امکان پیاده‌سازی این روش‌ها روی سیستم‌های تعبیه‌شده را فراهم نموده است. البته استفاده از این روش‌ها چالش‌های خاص خود همچون نیاز به تعدادی زیادی داده برای آموزش و ارزیابی سیستم، و نیاز به مهارت برای تنظیم پارامترهای شبکه را نیز به همراه دارد که مستلزم صرف زمان قابل توجه است

اعلام هشدار پس از تشخیص احتمال بروز خطر، وظیفه‌ی بعدی است که سامانه مورد نظر بایستی انجام دهد. با توجه به رواج استفاده از گوشی‌های هوشمند و برنامه‌های پیامرسان اینترنتی، و امکان ارسال پیام‌های چندرسانه‌ای در این پیامرسان‌ها، اعلام هشدار به صورت جمعی به افراد مسئول از طریق یک پیامرسان، مناسب و کافی به نظر می‌رسد، لازم به ذکر است گرچه رخداد الگوهای غیرعادی انقباض قلب می‌تواند نشانه خطرناکی باشد، اما از زمان شروع این وضعیت، در بدترین حالت که منجر به ایست قلبی شود، هنوز ۴ دقیقه برای رسیدگی و انجام عملیات احیای قلبی فرصت در اختیار کادر درمانی قرار دارد.

نمودار جریان داده سیستم

برای نمایش بهتر عملکرد کلی و درک سریع‌تر آن، نمودار جریان داده در ادامه آورده شده است:



تصویر۱: نمودار جریان - داده سیستم

بخش ۲:
مدل

۲-۱: دیتاست مورد استفاده

شامل ۴۵۰۰ رکورد دیتای آموزش و ۵۰۰ رکورد دیتای ECG برای انجام این پروژه از دیتاست ۵۰۰۰ ارزیابی است استفاده میکنیم. هر رکورد این دیتاست، شامل یک نمونه برداری با نرخ ۱۴۰ بار در دوره تناوب پالس سیگنال ECG و نیز برچسب آن رکورد (سیگنال نرمال و سیگنال ناهنجار) می‌باشد.

۲-۲: پیش پردازش داده‌ها

برای پیش پردازش داده ها، ابتدا ویژگی ها و برچسب داده ها را جدا میکنیم، سپس با استفاده از min max scaler تمامی داده ها را نرمال سازی میکنیم و به دو بخش داده تست و آموزش تقسیم بندی می‌کنیم.

۲-۳: الگوریتم مورد استفاده

با وجود تنوع بسیار بالای الگوریتم‌های نرمال پزشکی و یادگیری ماشین برای تشخیص ناهنجاری در سیگنال‌های ECG، در این پروژه از یک مدل، encoder-decoder استفاده کرده‌ایم. بخش encoder این مدل،شامل سه لایه fully connected با تعداد لایه های به ترتیب ۳۲، ۲۶ و ۸ می‌باشد و برای تابع activation از تابع relu استفاده کرده‌ایم. بخش decoder نیز، مشابه بخش encoder از سه لایه fully connected با تعداد لایه‌های به ترتیب ۱۶، ۳۲ و ۱۴۰ تشکیل شده است که در واقع از روی خروجی encoder، سیگنال ECG مشابه را بازسازی می‌کند. طبق روال کار شبکه‌های encoder-decoder، پس از بازسازی سیگنال ورودی، مقدار خطای ایجاد شده نسبت به سیگنال اولیه را با روش MAE(Mean Absolute Error) محاسبه می‌کنیم. اگر مقدار محاسبه شده از یک حد مشخص که آن را **آستانه سطح** می‌نامیم و مقدار آن در پروژه برابر ۰.۰۳۲ در نظر گرفته شده است، احتمال می‌دهیم که سیگنال بازسازی شده یک سیگنال ناهنجار است. نتایج بدست امده برای داده test در شبکه تولید شده به شکل زیر است:

```
preds = predict(autoencoder, test_data, 0.032)
print_stats(preds, test_labels)
```

```
Accuracy = 0.967
Precision = 0.9925233644859813
Recall = 0.9482142857142857
```

تصویر۲ : نتایج بدست آمده برای داده test با اجرای الگوریتم

۲-۴: خروجی مدل برای اجرا بر روی سخت افزار

بعد از **train** کردن شبکه، با استفاده از کتابخانه **tensorflow** و زیر بسته **tflite**، مدل را به فایل **hex** تبدیل کرده و به فرم مناسب برای اجرا بر روی سخت افزار در می‌آوریم:

```
converter = tf.lite.TFLiteConverter.from_saved_model("my_model.hd5f")
tflite_model = converter.convert()

with open('model.tflite', 'wb') as f:
    f.write(tflite_model)
```

تصویر۳: تبدیل مدل برای اجرا بر روی سخت افزار

بخش ۳:

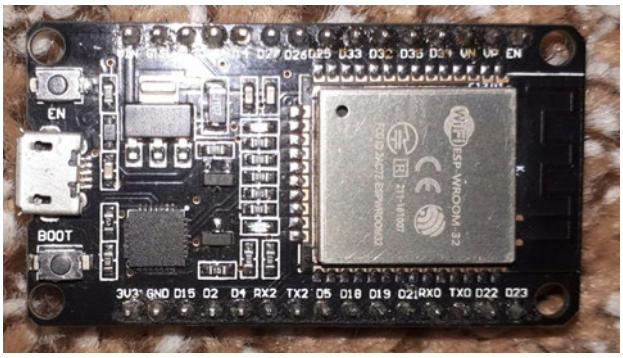
سخت افزار

۳-۱: مقدمه

پروژه حاضر دارای دو بخش اصلی سخت‌افزاری ماژول الکتروکاردیوگراف و ماژول ESP32 است، که ماژول الکتروکاردیوگراف وظیفه برداشت، تقویت و فیلتر کردن سیگنال‌های ماهیچه‌های قلب را بر عهده دارد و ماژول ESP32 موظف به تبدیل داده‌های آنالوگ دریافت‌شده از الکتروکاردیوگراف به داده‌های دیجیتال، بررسی این داده‌ها با شبکه‌عصبی، و ارسال پیام هشدار در مواقع بحرانی از طریق وای‌فای به سرور است.

۳-۲: ماژول ESP32

این ماژول در واقع برد توسعه مربوط به ESP-WROOM-32 یک SoC است که دارای پردازنده ۳۲بیتی دوهسته‌ای Xtensa LX6، مبدل آنالوگ به دیجیتال ۲کانال ۱۲بیتی با حداکثر نرخ نمونه برداری 2MHz، برقراری ارتباط از راه بلوتوث، و وای‌فای، سخت‌افزار مخصوص راه اندازی حسگر لمسی، حسگر اثر هال، UART،I2S،I2C، است که آن را به یک انتخاب مناسب برای کاربردهای اینترنت‌اشیا و سیستم‌های تعبیه‌شده تبدیل کرده است. برای راه‌اندازی و برنامه‌نویسی این برد، گزینه‌های مختلفی وجود دارد که در ادامه به توضیح آن‌ها پرداخته می‌شود:



تصویر ۴: ماژول ESP32

۳-۲-۱: Arduino IDE

استفاده از این محیط نسبتا راحت‌تر از سایر روش‌هاست، و دارای منابع و مثال‌های انجام‌شده فراوان است، و می‌توان با سرعت خوبی برد را برای کاربرد مورد نظر راه‌اندازی نمود، اما برای کاربرد حاضر، چون نیاز به استفاده از کتابخانه tflite برای پیاده‌سازی شبکه عصبی وجود دارد و کتابخانه موجود تنها برای نوع خاصی از بردهای آردوینو قابل استفاده است، امکان استفاده از این محیط در پروژه حاضر وجود نداشت.

۳-۲-۲: ESP-IDF

این محیط توسط شرکت سازنده برد توسعه داده شده و امکان دسترسی به تمام امکانات برد با استفاده از API مخصوص آن را فراهم می‌کند، اما یادگیری و استفاده از آن مستلزم صرف زمان زیاد است و از طرفی، اقبال عمومی نسبت به آن کم است و بنابراین، منابع کمتری درباره آن موجود است.

۳-۲-۳: افزونه ESP-IDF در vscode

این گزینه، با توجه به استفاده از محیط vscode، نسبت به خود ESP-IDF رابط کاربری بهتری دارد و از طرفی منابع بیشتری هم برای آن موجود است، اما همچنان چون بایستی از دستورات مربوط به ESP-IDF استفاده شود، که دارای پیچیدگی است.

۳-۲-۴: افزونه platformIO در vscode

این افزونه، با توجه به این که هم از محیط vscode، استفاده می‌نماید و هم قابلیت توسعه کد با هر دو دسته دستورات ESP-IDF، و Arduino IDE، را داراست، و برای کاربرد حاضر، مناسب‌ترین انتخاب است و امکان استفاده از چندین پلفترم در یک پروژه را برقرار می‌کند.

۳-۳: آماده سازی شبکه عصبی برای اجرا بر روی سخت افزار

برای این که مدل شبکه عصبی آموزش دیده‌شده را بتوانیم روی سخت‌افزار اجرا نماییم، نیاز است که داده‌های مدل به یک فایل زبان ماشین تفسیر شده و سپس با استفاده از کتابخانه tflite و توابع مربوط به آن فراخوانی و استفاده شود.

در کد زیر، ابتدا با استفاده از داده‌های استخراج‌شده از مدل اصلی، یک مدل ساخته شده و پس از آن حل‌کننده عملگرهای مختلف ساخته می‌شود و عملگرهایی که مدل ما استفاده می‌کند اضافه می‌شوند. در ادامه یک فضای کاری برای اجرای مفسر tflite در نظر گرفته شده و سپس مفسر واقعی ساخته می‌شود. پس از ساخت مفسر، شناسایی کند. در کد بالا، ابتدا با ذخیره کردن مشخصات نقطه دسترسی، اقدام به اتصال به آن می‌شود و در صورت موفقیت در اتصال، یک شی HTTP ساخته شده که امکان اجرای درخواست‌های GET، و POST برای آن وجود دارد. در کاربرد حاضر، ما عملا نیاز به استفاده از POST، داریم، و با توجه به نوع سروری که طراحی شده، بایستی برای آن یک شی داده json ارسال نماییم.

```
#include "NeuralNetwork.h"
#include "model_data.h"
#include "tensorflow/lite/micro/all_ops_resolver.h"
#include "tensorflow/lite/micro/micro_error_reporter.h"
#include "tensorflow/lite/micro/micro_interpreter.h"
#include "tensorflow/lite/schema/schema_generated.h"
#include "tensorflow/lite/version.h"

const int kArenaSize = 20000;

NeuralNetwork::NeuralNetwork()

{

    error_reporter = new tflite::MicroErrorReporter();

    model = tflite::GetModel(converted_model_tflite);

    if (model->version() != TFLITE_SCHEMA_VERSION)

    {

        TF_LITE_REPORT_ERROR(error_reporter, "Model provided is schema version %d not equal to
supported version %d.",
                             model->version(), TFLITE_SCHEMA_VERSION);

        return;

    }

    resolver = new tflite::MicroMutableOpResolver<10>();

    resolver->AddFullyConnected();

    resolver->AddMul();

    resolver->AddAdd();

    resolver->AddLogistic();

    resolver->AddReshape();

    resolver->AddQuantize();

    resolver->AddDequantize();

    tensor_arena = (uint8_t *)malloc(kArenaSize);
```

```
    if (!tensor_arena)

    {

        TF_LITE_REPORT_ERROR(error_reporter, "Could not allocate arena");

        return;

    }

    interpreter = new tflite::MicroInterpreter(
        model, *resolver, tensor_arena, kArenaSize, error_reporter);

    TfLiteStatus allocate_status = interpreter->AllocateTensors();
    if (allocate_status != kTfLiteOk)
    {
        TF_LITE_REPORT_ERROR(error_reporter, "AllocateTensors() failed");
        return;
    }

    size_t used_bytes = interpreter->arena_used_bytes();
    TF_LITE_REPORT_ERROR(error_reporter, "Used bytes %d\n", used_bytes);

    input = interpreter->input(0);
    output = interpreter->output(0);
}

float *NeuralNetwork::getInputBuffer()
{
    return input->data.f;
}

float NeuralNetwork::predict()
{
    interpreter->Invoke();
    return output->data.f[0];
}
```

۳-۴: اتصال به وای‌فای و برقراری ارتباط با سرور

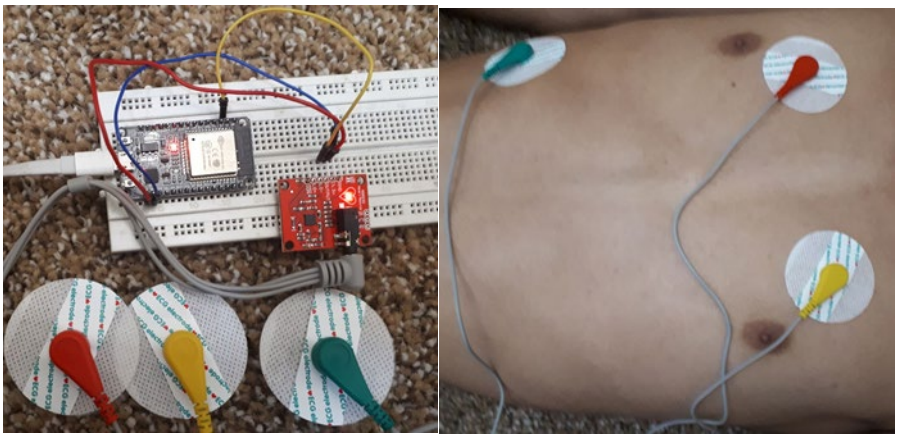
برای اتصال به وای‌فای توابع کتابخانه WiFi.h که در محیط Arduino و برای این سخت‌افزار قابل استفاده است به کار برده شده است و برای برقراری ارتباط نیز از کتابخانه HTTPClient.h بهره برده‌ایم که کد مربوط به آن در زیر آورده شده است:

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

const char* ssid = "AK";
const char* password = "aaaaaaaaaa";
const char* serverName = "http://xxx.xxx.xxx.xxx:xxxx/ecg/report/";

void setup()
{
  Serial.begin(115200);
    WiFi.begin(ssid, password);
  Serial.println("Connecting");
  while(WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to WiFi network with IP Address: ");
  Serial.println(WiFi.localIP());

      if(WiFi.status()== WL_CONNECTED){
      WiFiClient client;
      HTTPClient http;
      // Your Domain name with URL path or IP address with path
      http.begin(client, serverName);
      http.addHeader("Content-Type", "application/json");
      int httpResponseCode =
http.POST("{\"patient_number\":\"1\",\"alarm\":\"1\"}");
      Serial.print("HTTP Response code: ");
      Serial.println(httpResponseCode);
      http.end();
    }
    else {
      Serial.println("WiFi Disconnected");
    }
  }
  delay(1000);
}
```

در ابتدا لازم به ذکر است، برای اجرای کدهای آردوینو در محیط platformIO، لازم است ابتدا کتابخانه Arduino.h اضافه شود تا vscode که در واقع زبان c/c++ را پشتیبانی می‌کند، دستورات آردوینو را شناسایی کند. در کد بالا، ابتدا با ذخیره کردن مشخصات نقطه دسترسی، اقدام به اتصال به آن می‌شود و در صورت موفقیت در اتصال، یک شی HTTP ساخته شده که امکان اجرای درخواست‌های GET، و POST برای آن وجود دارد. در کاربرد حاضر، ما عملا نیاز به استفاده از POST، داریم، و با توجه به نوع سروری که طراحی شده، بایستی برای آن یک شی داده json ارسال نماییم.

۵-۳: دریافت داده‌های حسگر ECG

حسگر مورد استفاده برای دریافت داده‌های انقباض عضلانی قلب، دارای ۳ الکترود برای اتصال به بدن فرد بیمار است و خروجی آنالوگ 0-3.3V تولید می‌کند که بایستی توسط مبدل دیجیتال به آنالوگ دریافت شود، با توجه به شبکه عصبی طراحی‌شده، نیاز است که در هر ثانیه ۱۴۰داده به شبکه داده شود که به معنای داده‌برداری با فواصل زمانی ۷میلی‌ثانیه‌ای است که بدین منظور در یک حلقه تکرار، هر بار این ۱۴۰ داده جمع‌آوری شده و به عنوان ورودی به شبکه عصبی داده می‌شود.



تصویر ۵: دریافت داده‌های حسگر ECG

```
int index = 0;
  for(index = 0; index < 140; index++){
    float sample = (analogRead(36)*6/4096)-4;
    nn->getInputBuffer()[index] = sample;
    delay(7);
  }
  float result = nn->predict();
```

بخش ۴:
نرم افزار

۴-۱: مقدمه

قسمت نرم افزار پروژه دو هدف دارد. اولین هدف تعبیه ی یک API برای سخت افزار است که بتواند اطلاعات آنالیز شده را برای سرور بفرستد. دومین هدف ساخت یک ربات تلگرامی برای فیلتر کردن اطلاعات کاربران هنگام جستجو است. چگونگی دستیابی به این اهداف در ادامه به تفصیل شرح داده خواهد شد. زبان مورد استفاده در این قسمت python و framework مورد استفاده Django است. پایگاه داده‌ی مورد استفاده از سه بخش بیمار، پزشک، و سابقه‌های گزارش وضعیت بیمار در هر لحظه تشکیل می‌شود. هر بیمار چندین سابقه در سیستم دارد و چندین پزشک نیز او را ویزیت می‌کنند. در اینجا به دلیل آن که تفاوتی بین دسترسی پزشک و پرستار نیست، این دو را معادل می‌گیریم. هر بخش یک گروه تلگرامی دارد که برای ارسال پیام ها به پزشکان آن بخش از آن استفاده می‌شود. افزودن پزشک و بیمار و ویرایش اطلاعات آن ها نیز توسط Django admin انجام می‌شود. DBMS مربوط به این پروژه نرم افزار postgresql است که باید از پیش در سیستم نصب شده باشد و یک پایگاه داده با مشخصاتی که در قسمت تنظیمات Django است در آن به وجود آید.

۴-۲: API

این قسمت، که توسط یک view در Django-rest-framework پیاده سازی می‌شود، به این صورت کار می‌کند که یک post request حاوی Json، که فرم آن در پایین نمایش داده می‌شود، به این url ارسال می‌شود: http://{server-address}/ecg/report/. برنامه این سابقه را در پایگاه داده ذخیره می‌کند و در صورت اعلام وجود مشکل قلبی از سمت سخت افزار یک پیام هشدار در گروه تلگرامی که پزشکان آن بخش در آن عضو هستند ارسال می‌کند.

فرم json:

```
{
    patient_number : {patient_number},
    alarm : {1 for unnormal, 0 for normal}
}
```

دلیل اینکه در بدنه درخواست مقدار patient_number را به صورت static قرار می‌دهیم این است که هر دستگاه، یک patient number مربوط به خود را دارد که منحصر به فرد است.

۴-۳: ربات تلگرام

این قسمت که توسط script های موجود در Django پیاده سازی شده است به این صورت عمل می‌کند که یک فیلتر برای جستجو در ربات تلگرامی تعبیه شده است. در اینجا دو قابلیت جستجو برای نمونه گذاشته شده است. این قابلیت ها برحسب نیاز بیمارستان قابل توسعه است. توضیحات زیر با زدن help/ در ربات تلگرام قابل دسترسی است. خروجی این جستجوها نیز در قالب یک فایل csv برای کاربر ارسال می‌شود.

قابلیت اول برای جستجو روی بیمار است. به این صورت که کاربر کد بیمار را در قالب /n/{patient_number} می‌فرستد. و همه ی record های مربوط به آن بیمار تا آن لحظه فرستاده می‌شود.

قابلیت دوم نیز سرچ روی بازه‌ی زمانی است. کاربر بازه‌ی زمانی را به این صورت می‌فرستد:

/t/jalali-jalali

برای مثال برای گرفتن کل رکورد های یک ماه اخیر دستور به این صورت فرستاده می‌شود:

/t/1400/11/11-1400/10/11

برای راحتی کاربر تاریخ ها به شمسی وارد می‌شود و در سرور به میلادی تبدیل می‌شود.

بخش ۵:

ویدیوهای بررسی صحت

۱. تست نرم افزار و ربات تلگرام
۲. تست کلی سخت افزار و نرم افزار

مراجع

1) atomic14/tensorflow-lite-esp32
2) ESP32 HTTP GET and HTTP POST with Arduino IDE (JSON, URL Encoded, Text)
3) ESP32: HTTP POST Requests
4) Dataset: ECG5000
5) pr1266/iot_ECG_signals_process

Isfahan University of Technology

College of Engineering

School of Computer Engineering

Intelligent embedded system for monitoring and analysis of electrocardiographic signals and warning of possible danger

Pouria Pour Yeganeh, Alireza Salehi Hossein Abadi

Alireza Kashti Ara, Mehdi Shirvani

February 2022