

CENTRO DE ENSEÑANZA TÉCNICA INDUSTRIAL
Ingeniería en Desarrollo de Software
Aseguramiento de la calidad en Software
FEBRERO – JULIO 2023



Herramientas de la calidad

Grado y grupo: 8P

Fecha de entrega: 15/02/2022

Nombre del docente:

Rosa María Santana Vázquez

Equipo:

Carlo Pinedo Suárez - 19310163

Contenido

Propuesta	3
Materiales	3
Diagramas de conexión	4
Raspberrry 3 pinout	4
ADC1115.....	4
Sensor de frecuencia cardiaca	4
OPAM 741	5
Conexión entre el adc y la raspberry.....	5
Conexión del sensor y el amplificador operacional 741	6
Desarrollo	6
Ajustando la señal del sensor.....	6
Configurando la raspberry	7
Código raspberry	13
Código php	17
Resultados	18
Señal obtenida del sensor	18
Señal medida en la raspberry.....	18
Imágenes obtenidas de la página web	19
Alertas enviadas	21

Propuesta

La idea es ser un proyecto cuyo impacto social está enfocado en el área médica, más específicamente a pacientes que se encuentran en cama, con un sensor de frecuencia cardiaca estarán monitoreando algunas características que se obtienen a través de esta medición, frecuencia como su nombre lo dice, si hay algún tipo de taquicardia también lo informaría o en su determinado defecto que el corazón deje de latir, etc.

Dichas características se estarían enviando constantemente actualizadas a un sistema ya sea web o móvil para que la persona que está cuidando del paciente este segura de lo que está pasando en todo momento, si llegara a ocurrir algo fuera de lo normal se enviaría una notificación de informe.

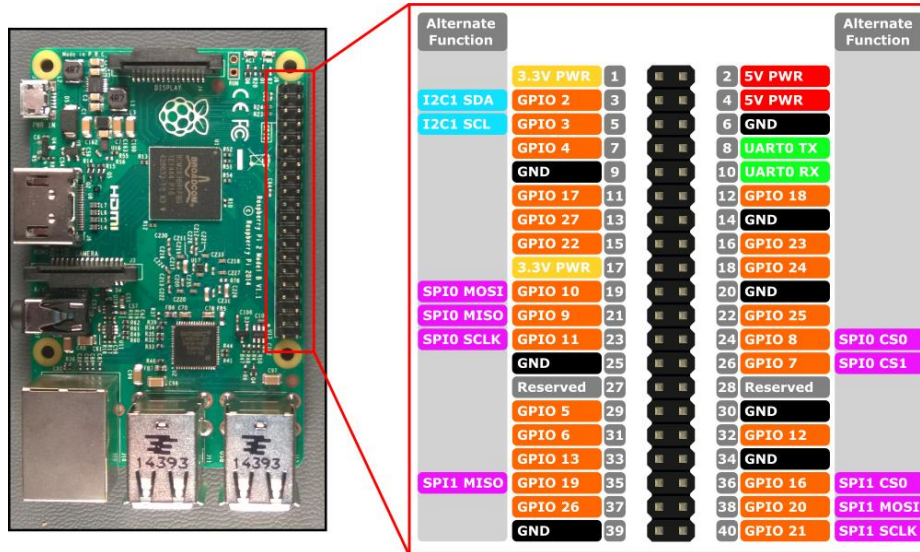
El servicio en la nube utilizado será Azure.

Materiales

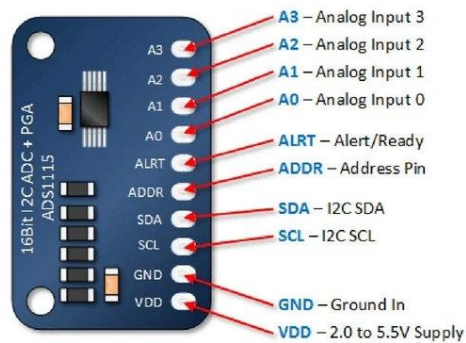
Componente	Cantidad
Raspberry pi 3	1
Sensor de frecuencia cardiaca Arduino	1
ADC 1115	1
OPAM 741	1
Resistencia 10k	1
Resistencia 4.7k	1
Capacitor 10uf	1
Caimanes	4 aprox
Cables dupont	15 aprox
Cable para proto	2mt aprox
Protoboard	1
Fuente de voltaje de +/-10v	1
Osciloscopio(opcional)	1
Laptop para trabajar en el Código	1

Diagramas de conexión

Raspberry 3 pinout



ADC1115



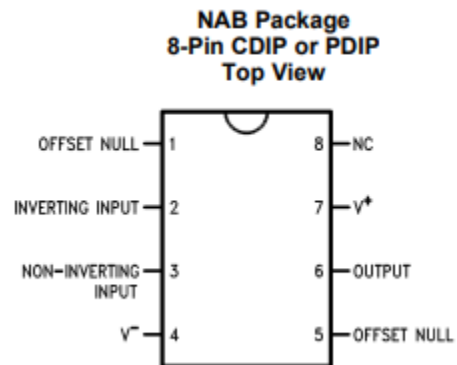
Sensor de frecuencia cardiaca

El cable negro del sensor representa la Tierra, el cable rojo el voltaje de alimentación lleva de 3v a 5v el cable rosado es el que habla que entrega los datos, en realidad no importa mucho el color sino el orden, ya que puedes comprar un sensor que tenga colores distintos sin embargo el orden siempre será el mismo.



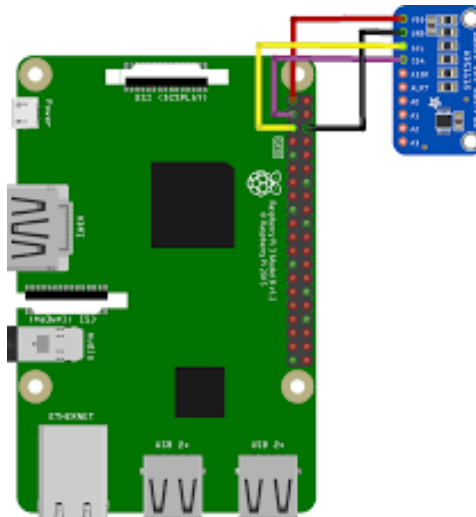
OPAM 741

La señal que nos entregue el sensor de frecuencia cardíaca es muy pequeña para esto es necesario amplificarla y esto se hará con el amplificador operacional 741, más adelante esperemos o conexión. Evidentemente para que este pueda funcionar es necesario alimentarlo con un voltaje de $\pm 10\text{v}$.



Conexión entre el adc y la raspberry

La conexión aquí no es muy complicada sólo hay que fijarse en dónde están los pines de I2C en la raspberry, el adc usa un voltaje de 3.3v entonces perfectamente podemos conectarlo a la salida de voltaje de la tarjeta.

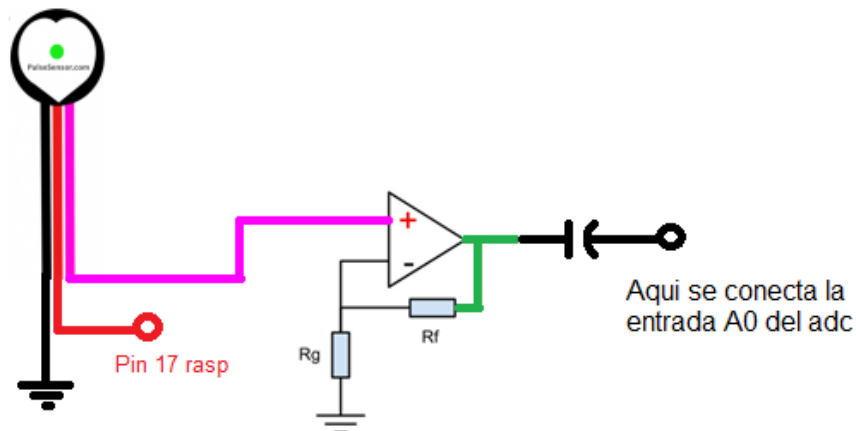


Conexión del sensor y el amplificador operacional 741

La configuración del 741 será de amplificador no inversor, esto porque únicamente queremos que la señal del sensor se haga más grande para que el adc pueda leerla, el capacitor que vemos ahí conectado es el de 10uF que vemos en la lista de materiales, la idea de ese es evitar cualquier tipo de offset causado por una corriente directa no deseada.

$$R_f = 10k\Omega$$

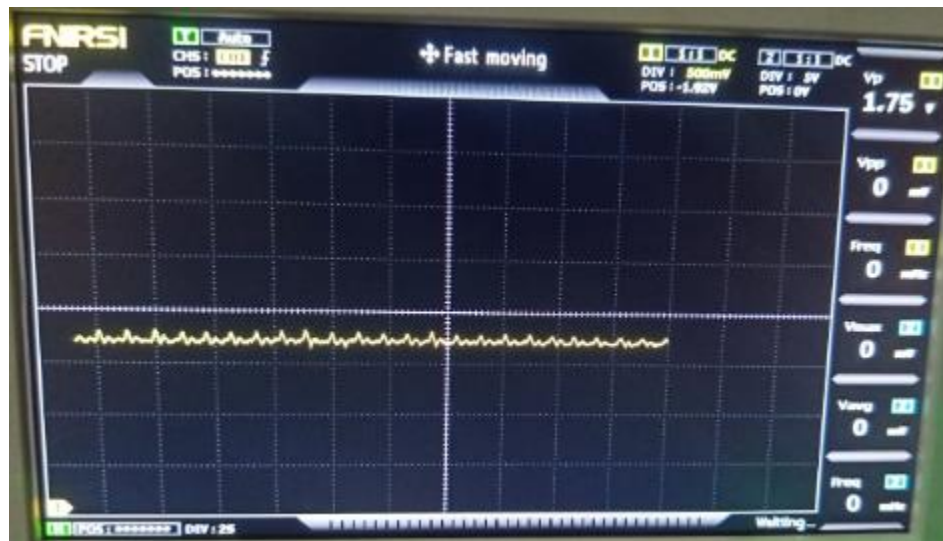
$$R_g = 4.7k\Omega$$



Desarrollo

Ajustando la señal del sensor

La señal que nos da el sensor por sí solo es muy débil como vemos en la siguiente imagen, y por sí sola no es suficiente para que nuestro adc la detecte.



Es por eso que es necesario amplificarla, yo utilicé el amplificador operacional 741, Que como vemos en el diagrama tiene 2 resistencias que nos ayudarán a obtener la amplificación, la ecuación utilizada es la siguiente dónde S es la salida y E es la señal de entrada, lo que vemos dentro de los paréntesis es básicamente la ganancia:

$$S = E\left(1 + \frac{R_f}{R_g}\right)$$

Una vez amplificada a la señal, obtuve lo siguiente y como vemos ha crecido en voltaje y con esto es suficiente para el adc.



Configurando la raspberry

En este punto ya es factible empezar a realizar las configuraciones la raspberry, es importante aclarar que yo estoy programando la tarjeta a través del ssh de la laptop con un cable, previamente tuve que activar el ssh de la raspberry y configurar el Wi-Fi.

Antes de poder conectar el ADC con el que obtendremos la señal del sensor es necesario activar el I2C de nuestra tarjeta.

Para esto ejecutamos el siguiente comando en la consola, obviamente de la tarjeta.

```
sudo apt-get install -y python-smbus
```

```
sudo apt-get install -y i2c-tools
```

Una vez instalados los paquetes ejecutamos el siguiente comando

```
sudo raspi-config
```

Y nos saldrá una pantalla como la siguiente, debemos ir a opciones de interfaz:

```
Raspberry Pi Software Configuration Tool (raspi-config)

1 Change User Password      Change password for the default u
2 Hostname                  Set the visible name for this Pi
3 Boot Options              Configure options for start-up
4 Localisation Options      Set up language and regional sett
5 Interfacing Options       Configure connections to peripher
6 Overclock                 Configure overclocking for your P
7 Advanced Options         Configure advanced settings
8 Update                   Update this tool to the latest ve
9 About raspi-config        Information about this configurat

<Select>                  <Finish>
```

Luego vamos a opciones avanzadas.

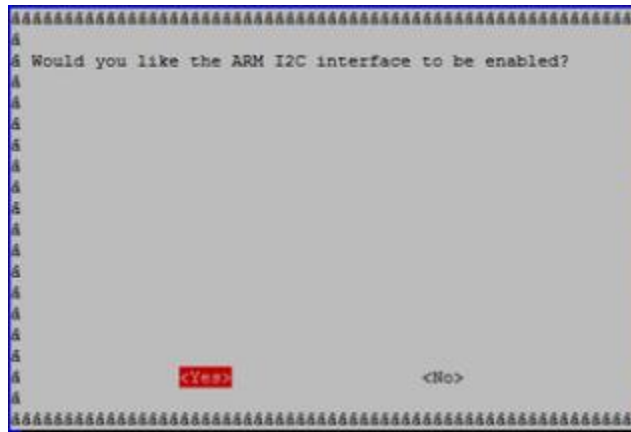
```
##### Raspberry Pi Software Configuration Tool (raspi-config) #####
1 Expand Filesystem        Ensures that all of the SD card storage is availab
2 Change User Password     Change password for the default user (pi)
3 Enable Boot to Desktop/Scratch Choose whether to boot into a desktop environment,
4 Internationalisation Options Set up language and regional settings to match you
5 Enable Camera            Enable this Pi to work with the Raspberry Pi Camer
6 Add to Rastrack          Add this Pi to the online Raspberry Pi Map (Rastra
7 Overclock                Configure overclocking for your Pi
8 Advanced Options         Configure advanced settings
9 About raspi-config       Information about this configuration tool

<Select>                  <Finish>
```

Elegimos I2C y lo habilitamos.

```
##### Raspberry Pi Software Configuration Tool (raspi-config) #####
A1 Overscan               You may need to configure overscan if black bars are present on display
A2 Hostname               Set the visible name for this Pi on a network
A3 Memory Split           Change the amount of memory made available to the GPU
A4 SSH                    Enable/Disable remote command line access to your Pi using SSH
A5 Device Tree            Enable/Disable the use of Device Tree
A6 SPI                    Enable/Disable automatic loading of SPI kernel module (needed for e.g. PiFace)
A7 I2C                    Enable/Disable automatic loading of I2C kernel module
A8 Serial                 Enable/Disable shell and kernel messages on the serial connection
A9 Audio                  Force audio out through HDMI or 3.5mm jack
A0 Update                 Update this tool to the latest version

<Select>                  <Back>
```

Ejecutamos y con esto podemos proceder a lo siguiente

```
sudo reboot
```

Ahora sí, para poder utilizar el ADC1115 es necesario instalar las librerías esto se hace con los siguientes comandos los cuales insertaremos en la consola de la raspberry.

```
sudo apt-get update sudo apt-get install build-essential python-dev  
python-smbus git cd ~ git clone  
https://github.com/adafruit/Adafruit\_Python\_ADS1x15.git cd  
Adafruit_Python_ADS1x15 sudo python setup.py install
```

También instalaremos el índice de paquetes de Python.

```
sudo apt-get update sudo apt-get install build-essential python-dev  
python-smbus python-pip sudo pip install adafruit-ads1x15
```

Hasta aquí ya deberíamos tener todo listo para poder recibir la señal de nuestro sensor en la raspberry y poder leer los datos que más adelante eran enviados a nuestro servicio de iot.

Configurando Azure

Una vez que se tiene una cuenta, es necesario crear un grupo de recursos, este no servirá para guardar en grupos los recursos que vamos a utilizar.

Al querer utilizar un nuevo recurso nos pedirá que seleccionemos un grupo y una vez seleccionado se podrá transmitir información entre recursos que estén en el mismo grupo.

En mi caso el grupo utilizado se llama PROYECTO.

Microsoft Azure | Buscar recursos, servicios y documentos (G+/)

Inicio > Grupos de recursos

Centro de Enseñanza Técnica Industrial (live.ceti.mx)

+ Crear | Administrar vista | Actualizar | Exportar a CSV | Abrir consulta | Asignar etiquetas

Filtrar por cualquier ca... | Suscripción es igual a todo | Ubicación es igual a todo | + Agregar filtro

0 Recursos no seguros | 0 Recomendaciones | Sin agrupar | Vista de lista

Nombre ↑↓	Suscripción ↑↓	Ubicación ↑↓
cloud-shell-storage-southcentralus	Azure subscription 1	South Central US
DefaultResourceGroup-CUS	Azure subscription 1	Central US
DefaultResourceGroup-EUS	Azure subscription 1	East US
PROYECTO	Azure subscription 1	West US

Lo que sigue es crear un centro de iot, este tendrá como finalidad darnos la capacidad de conectar nuestra raspberry con los servicios de Asure, básicamente aquí creamos un nuevo dispositivo que contendrá información de conexión para nuestra tarjeta.

Microsoft Azure | Buscar recursos, servicios y documentos (G+/)

Inicio > proyecto-tecnologias-emergentes

proyecto-tecnologias-emergentes | Dispositivos

Centro de IoT

Buscar | Consulte, cree, elimine y actualice los dispositivos de su instancia de IoT Hub. Más información

+ Agregar dispositivo | Actualizar | Asignar etiquetas | Eliminar | Búsqueda de dispositivos mediante una consulta

escribir el id. de dispositivo | Tipos: todos | + Agregar filtro

Identidad del dispositivo	Tipo	Estado	Última actualización de est...	Tipo de autenticac...	Mens...	Etique...
rasperry	Dispositivo IoT	Habilitado	--	Firma de acceso co...	0	

Parte de la información que vemos en la siguiente imagen deberemos copiarla en nuestro código de la raspberry.

Microsoft Azure | Buscar recursos, servicios y documentos (G+/)

Inicio > proyecto-tecnologias-emergentes | Dispositivos >

rasperry

proyecto-tecnologias-emergentes

Guardar | Mensaje al dispositivo | Método directo | + Agregar identidad de módulo | Dispositivo gemelo | Actualizar

Id. de dispositivo: rasperry

Clave principal: [oculto]

Clave secundaria: [oculto]

Cadena de conexión principal: [oculto]

Cadena de conexión secundaria: [oculto]

Etiquetas: Sin etiquetas

Habilitar la conexión a IoT Hub: ☒ Habilitar ☐ Deshabilitar

Dispositivo principal: No hay ningún dispositivo primario

Después hay que crear la base de datos donde se almacenarán los datos que nos envíe la raspberry el sensor.

Microsoft Azure portal showing the 'SQL Database' page. The page displays a table with one database instance named 'pulsoCardiacodb'.

Nombre	Servidor	Tipo de réplica	Plan de tarifa	Ubicación	Suscripción
pulsoCardiacodb (iotedu/pulsoCardiacodb)	iotedu	--	Standard S0: 10 DT...	East US	Azure subscription 1

Como podemos ver en la imagen yo creé una tabla con 3 columnas id, frecuencia y fecha que me servirán más adelante mi sistema.

Microsoft Azure portal showing the 'pulsoCardiacodb' database editor. The editor displays a table with 3 columns: id, frecuencia, and fecha.

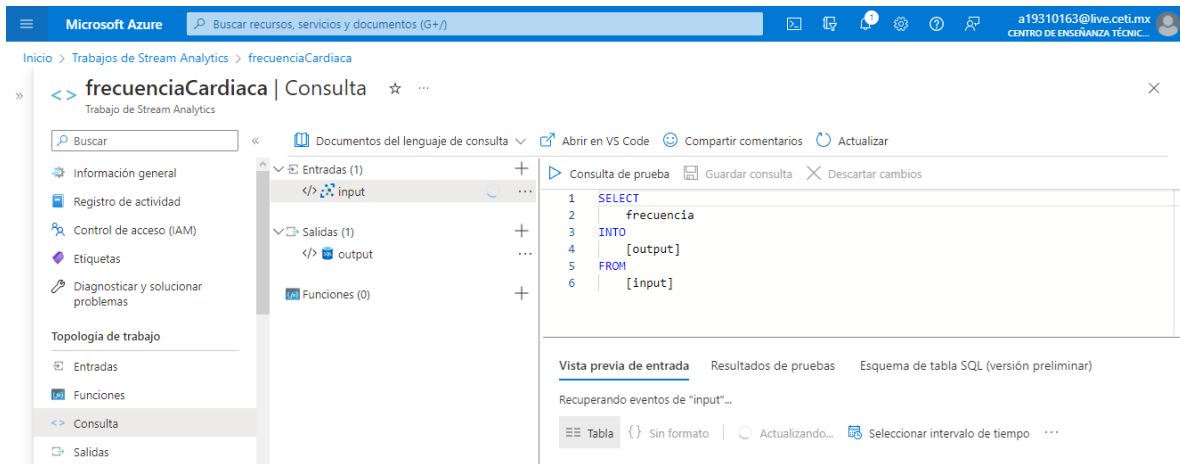
id	frecuencia	fecha
88.2742386346918	46	2023-02-15T01:36:47.3630000
87.3235336923301	47	2023-02-15T01:36:47.3630000
87.8605945233563	48	2023-02-15T01:36:48.5670000
88.0023467292461	49	2023-02-15T01:36:48.5670000

Para poder enviar los datos de nuestra raspberry a nuestra base de datos en tiempo real es necesario el recurso de stream, este lo que hará será constantemente recibir los datos que envíe la tarjeta y enviarlos a nuestra base de datos.

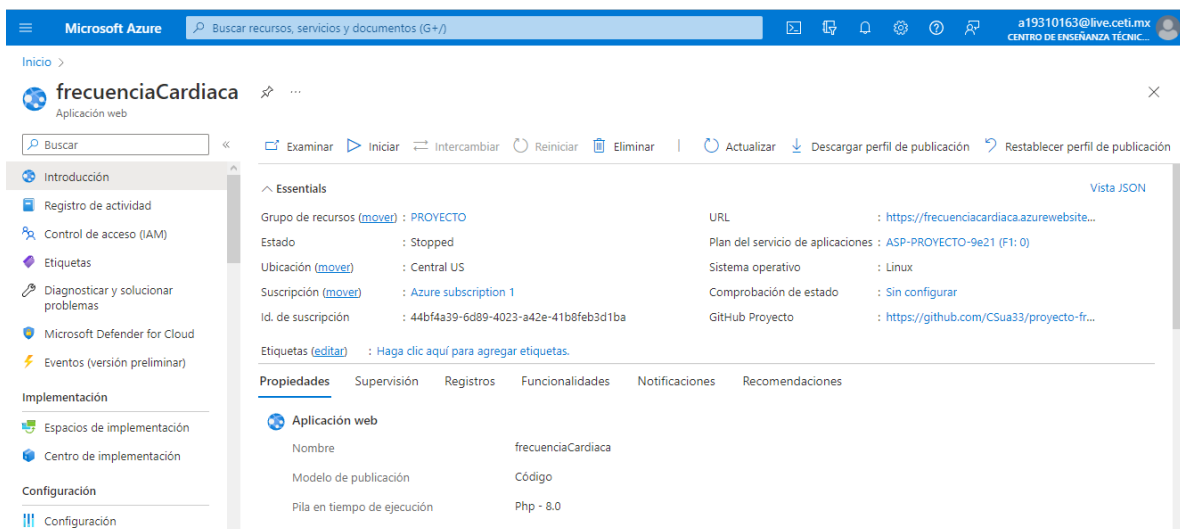
Microsoft Azure portal showing the 'Trabajos de Stream Analytics' page. The page displays a table with 2 stream jobs: 'frecuenciaCardiaca' and 'Stream'.

Nombre	Grupo de recursos	Ubicación	Estado	Tipo	Nivel de compatibi...	Creado (UTC)	Marca de agua de salida (U...
frecuenciaCardiaca	PROYECTO	East US	Stopped	Cloud	1.2	2023-02-14 04:13:48	2023-02-15 01:43:51
Stream	PROYECTO	East US	Stopped	Cloud	1.2	2022-12-31 00:24:38	2023-02-14 05:22:10

En la siguiente imagen podemos ver que tengo un input y un output, el input son los datos que vienen de la raspberry y el output, la consulta que se hace para enviar los datos a la base de datos.



Finalmente esto yo lo hice por probar el servicio de hosting de una página web de Azure, utilice el servicio llamado aplicación web para poder hostear mi sistema, lo interesante de este servicio es que puedes subir tu página desde GITHUB.



Código raspberry

Una vez que ya tenemos todas las configuraciones necesarias podemos proceder a programar y el primer código que voy a mostrar es el de la raspberry.

Lo primero que vamos a ver en este código es que estoy importando unas librerías las más importantes son las del adc y la de azure, La primera librería que menciono nos da las herramientas para poder convertir de manera fácil la señal analógica entregada por el sensor a una señal digital que la raspberry pueda leer y la segunda librería nos da las funciones necesarias para poder enviar datos a azure.

Todo lo demás viene bastante explicado y la idea es simplemente darle un valor de BPM a la señal analógica que leemos para luego enviarla a nuestro servicio de iot.

```
import time
# Import the ADS1x15 module.
import Adafruit_ADS1x15
from azure.iot.device import IoTHubDeviceClient, Message

# az iot hub device-identity show-connection-string --hub-name
{YourIoTHubName} --device-id MyNodeDevice --output table
CONNECTION_STRING = "HostName=proyecto-tecnologias-emergentes.azure-
devices.net;DeviceId=raspberry;SharedAccessKey=e0Fj6comkvgNp3JBRG1P7+csUB8i/i
Um18HYfEt7oeU="

# Define the JSON message to send to IoT Hub.
MSG_TXT = '{{"frecuencia": {frecuencia}}}'

#Choose a gain of 1 for reading voltages from 0 to 4.09V.
# Or pick a different gain to change the range of voltages that are read:
# - 2/3 = +/-6.144V
# - 1 = +/-4.096V
# - 2 = +/-2.048V
# - 4 = +/-1.024V
# - 8 = +/-0.512V
# - 16 = +/-0.256V
# See table 3 in the ADS1015/ADS1115 datasheet for more info on gain.

if __name__ == '__main__':

    client =
IoTHubDeviceClient.create_from_connection_string(CONNECTION_STRING)
    print ( "IoT Hub device sending periodic messages, press Ctrl-C to exit"
)

    adc = Adafruit_ADS1x15.ADS1115()
    # initialization
    GAIN = 8
```

```

curState = 0
thresh = 525 # mid point in the waveform
P = 512
T = 512
stateChanged = 0
sampleCounter = 0
lastBeatTime = 0
firstBeat = True
secondBeat = False
Pulse = False
IBI = 600
rate = [0]*10
amp = 100

lastTime = int(time.time()*1000)

# Main loop. use Ctrl-c to stop the code
while True:
    # read from the ADC
    Signal = adc.read_adc(0, gain=GAIN) #TODO: Select the correct ADC
    channel. I have selected A0 here
    curTime = int(time.time()*1000)

    sampleCounter += curTime - lastTime; # # keep
    track of the time in mS with this variable
    lastTime = curTime
    N = sampleCounter - lastBeatTime; # # monitor the time since
    the last beat to avoid noise
    #print N, Signal, curTime, sampleCounter, lastBeatTime

    ## find the peak and trough of the pulse wave
    if Signal < thresh and N > (IBI/5.0)*3.0 : # # avoid
    dichrotic noise by waiting 3/5 of last IBI
        if Signal < T : # T is the trough
            T = Signal; # keep track of lowest
            point in pulse wave

        if Signal > thresh and Signal > P: # thresh condition
        helps avoid noise
            P = Signal; # P is the peak
            # keep track of highest
            point in pulse wave

    # NOW IT'S TIME TO LOOK FOR THE HEART BEAT
    # signal surges up in value every time there is a pulse

```

```

        if N > 250 :                                # avoid high
frequency noise
        if (Signal > thresh) and (Pulse == False) and (N >
(IBI/5.0)*3.0) :
            Pulse = True;                            # set the Pulse
flag when we think there is a pulse
            IBI = sampleCounter - lastBeatTime;        # measure time
between beats in mS
            lastBeatTime = sampleCounter;            # keep track of
time for next pulse

            if secondBeat :                            # if this is the second
beat, if secondBeat == TRUE
                secondBeat = False;                    # clear secondBeat flag
                for i in range(0,10):                # seed the running total
to get a realisitc BPM at startup
                    rate[i] = IBI;

            if firstBeat :                            # if it's the first time
we found a beat, if firstBeat == TRUE
                firstBeat = False;                    # clear firstBeat flag
                secondBeat = True;                    # set the second beat
flag
                continue                                # IBI value is
unreliable so discard it

            # keep a running total of the last 10 IBI values
            runningTotal = 0;                        # clear the runningTotal
variable

            for i in range(0,9):                    # shift data in the rate
array
                rate[i] = rate[i+1];                    # and drop the oldest
IBI value
                runningTotal += rate[i];                # add up the 9 oldest
IBI values

            rate[9] = IBI;                            # add the latest IBI
to the rate array
            runningTotal += rate[9];                # add the latest IBI
to runningTotal
            runningTotal /= 10;                    # average the last 10
IBI values

```

```

        BPM = 60000/runningTotal;                                # how many beats can
fit into a minute? that's BPM!
        #Here is the part that i will use to send data
        frecuencia=BPM
        msg_txt_formatted = MSG_TXT.format(frecuencia=frecuencia)
        message = Message(msg_txt_formatted)
        print(type(frecuencia))
        print( "Sending message: {}".format(message) )
        client.send_message(message)
        print ( "Message successfully sent" )
        print ( 'BPM: {}'.format(BPM))

        if Signal < thresh and Pulse == True :    # when the values are going
down, the beat is over
            Pulse = False;                        # reset the Pulse flag so
we can do it again
            amp = P - T;                          # get amplitude of the
pulse wave
            thresh = amp/2 + T;                   # set thresh at 50% of
the amplitude
            P = thresh;                           # reset these for next
time
            T = thresh;

        if N > 2500 :                                # if 2.5 seconds go by
without a beat
            thresh = 512;                          # set thresh default

        if N > 2500 :                                # if 2.5 seconds go by
without a beat
            thresh = 512;                          # set thresh default
            P = 512;                                # set P default
            T = 512;                                # set T default
            lastBeatTime = sampleCounter;           # bring the lastBeatTime
up to date
            firstBeat = True;                      # set these to avoid
noise
            secondBeat = False;                   # when we get the
heartbeat back
            print ("no beats found")

        time.sleep(0.005)

```


Código php

En esta parte no voy a mostrar todo el código porque si no me quedaría un documento bastante grande y poco entendible pero sí voy a mostrar cómo hago la conexión a la base de datos de azure que es de las partes relevantes del proyecto.

Para esto azure nos permite hacer la conexión en distintos lenguajes, yo estoy utilizando php, y básicamente lo que tenemos es una variable que nos guarda el string con los datos necesarios para la conexión, ahí yo lo único que tuve que modificar es qué base de datos está utilizando y los datos para acceder a dicha base de datos como usuario y contraseña.

una vez que la conexión se hizo correctamente ya puedo hacer mis consultas como vemos en la variable llamada \$tsql.

```
<?php
// PHP Data Objects(PDO) Sample Code:
try {
    $conn = new PDO("sqlsrv:server = tcp:iotedu.database.windows.net,1433;
Database = pulsoCardiacodb", "CloudSA1ce0f26e", "{contraseña}");
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
}
catch (PDOException $e) {
    print("Error connecting to SQL Server.");
    die(print_r($e));
}

// SQL Server Extension Sample Code:
$connectionInfo = array("UID" => "CloudSA1ce0f26e", "pwd" => "{Cafu2025.}",
"Database" => "pulsoCardiacodb", "LoginTimeout" => 30, "Encrypt" => 1,
"TrustServerCertificate" => 0);
$serverName = "tcp:iotedu.database.windows.net,1433";
$conn = sqlsrv_connect($serverName, $connectionInfo);
//echo "Hola mundo";
//echo "<br>";
    //echo "Connection was established";
    //echo "<br>";

    $tsql = "SELECT TOP 1 frecuencia FROM [dbo].[ejemplo] ORDER BY id DESC";
    $stmt = sqlsrv_query($conn, $tsql);
    if ($stmt === false) {
        echo "Error in query execution";
        echo "<br>";
        die(print_r(sqlsrv_errors(), true));
    }
    $row = sqlsrv_fetch_array($stmt, SQLSRV_FETCH_ASSOC);
    $data = $row['frecuencia'];?>
```

Resultados

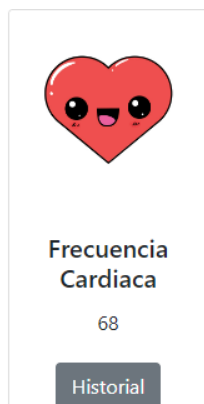
Señal obtenida del sensor



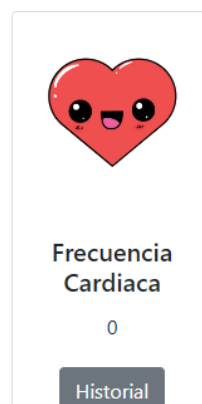
Señal medida en la raspberry

```
pi@raspberrypi: ~/Adafruit_Python_ADS1x15/examples
BPM: 68.46970215679562
BPM: 68.42285323297982
BPM: 68.50097043041443
BPM: 68.68918145392101
BPM: 68.75214850464077
BPM: 68.94174422612892
BPM: 69.06077348066299
BPM: 68.9496667432774
BPM: 68.5949468389162
BPM: 68.36827711941659
BPM: 68.25162097599818
BPM: 68.25938566552901
BPM: 68.6027898467871
BPM: 68.84681583476764
BPM: 68.96551724137932
BPM: 69.20415224913495
BPM: 69.44444444444444
BPM: 69.67831842991522
BPM: 69.68641114982579
BPM: 70.16723190270143
BPM: 70.3152466893238
BPM: 70.17543859649123
BPM: 72.59528130671507
BPM: 75.60483870967742
```

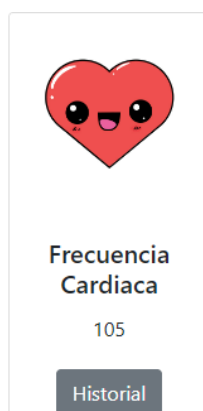
Imágenes obtenidas de la página web



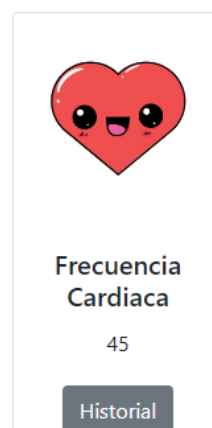
Paciente estable



Alerta: PELIGRO

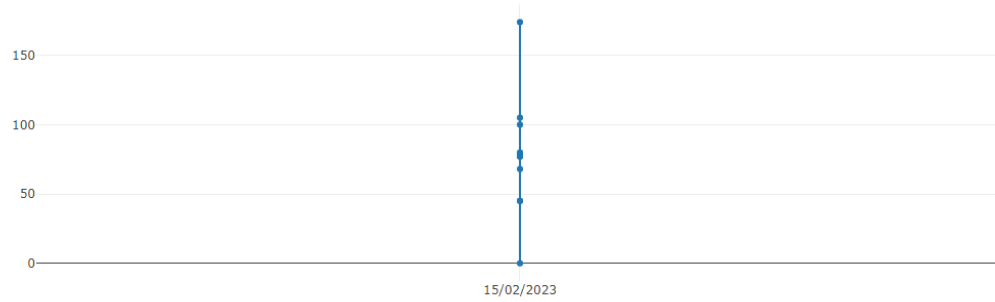


Alerta: taquicardia

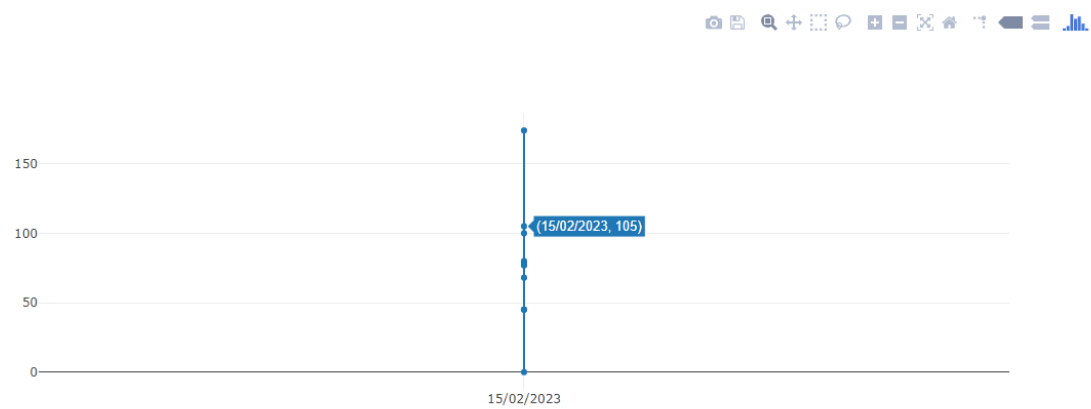


Alerta: bradicardia

Grafica en el tiempo de la frecuencia cardiaca



Grafica en el tiempo de la frecuencia cardiaca



Alertas enviadas

1-50 de 2,079

Principal

Promociones 2 nuevas
The Ionic Team, DiDi Food

Social 2 nuevas
Twitter

<input type="checkbox"/>	☆ yo	Alerta: el paciente se muere - Alerta: el paciente se muere	14 feb
<input type="checkbox"/>	☆ yo	Alerta: el paciente se muere - Alerta: el paciente se muere	14 feb
<input type="checkbox"/>	☆ yo	Alerta: el paciente se muere - Alerta: el paciente se muere	14 feb
<input type="checkbox"/>	☆ yo	Alerta: el paciente se muere - Alerta: el paciente se muere	14 feb
<input type="checkbox"/>	☆ yo	Alerta: el paciente se muere - Alerta: el paciente se muere	14 feb
<input type="checkbox"/>	☆ yo	Alerta: el paciente se muere - Alerta: el paciente se muere	14 feb
<input type="checkbox"/>	☆ yo	Alerta: el paciente se muere - Alerta: el paciente se muere	14 feb
<input type="checkbox"/>	☆ yo	Alerta: el paciente se muere - Alerta: el paciente se muere	14 feb
<input type="checkbox"/>	☆ yo	Alerta: el paciente se muere - Alerta: el paciente se muere	14 feb
<input type="checkbox"/>	☆ yo	Alerta: el paciente se muere - Alerta: el paciente se muere	14 feb
<input type="checkbox"/>	☆ yo	Alerta: el paciente se muere - Alerta: el paciente se muere	14 feb