UNIVERSITÉ VIRTUELLE DE TUNIS

MASTER IASRIA

MINI-PROJET ELECTRONIQUE

Détection des troubles cardiologiques à partir de l'analyse des ECG

Auteurs
R. ALOUI
N. AZZOUZ
A. MALLEH

Directeur
Dr. Rached GHARBI

2022 - 2023

Table des matières

Remerciements

> Si j'ai pu voir plus loin, c'est que je me tenais sur les épaules de géants.
>
> *Isaac Newton*

Nous tenons à exprimer toutes notre reconnaissance envers nos enseignants. Pour le sérieux, l'engagement et l'encadrement de M. **Rached GHARBI**, nous sommes fiers de passer sous votre direction.

Introduction générale

Les variétés de l'activité humaine aujourd'hui exige le suivi de l'état de santé à proximité. Ce qui s'est traduit par l'augmentation des équipements portés (wearable healthcare) de suivi du santé chaque année par 24% chaque année [1] [1]. La montre intélligente offre plusieurs services outre que le temps. La diversité de ces équipements prend plusieurs formes et assure la collecte de plusieurs métriques, la pression artérielle, concentration de O_2, ...

L'électrocardiogramme (ECG) est un test simple pour se renseigner rapidement sur le rythme et l'activité électrique du cœur. C'est pourquoi l'analyse d'une telle information critique aide à conclure sur la normalité de l'activité cardiaque d'une façon générale. L'analyse du signal ECG, selon le type d'analyse, peut se faire sur plusieurs étapes, le traitement du signal, l'extraction des caractéristiques, les transformations et finalement la classification [2, 3].

Aujourd'hui, la connectivité peut améliorer l'acquisition et l'enregistrement des données. C'est l'ère des objets connectés où ces objets peuvent envoyer les informatios sur des plateformes dédiées. Ces plateformes offrent d'autres traitements sur les données collectées.

Nous voulons, dans ce projet, concevoir un système de suivi de l'activité cardiaque. La solution doit être connectée. Le présent rapport détaille les étapes de conception et de réalisation de la solution.

1. https ://www.insiderintelligence.com/insights/wearable-technology-healthcare-medical-devices/

Chapitre 1

Description de la solution

1.1 Mise en cadre du projet

Rappelons que la solution doit :

1. assurer suivi de l'activité cardiaque,
2. effectuer sauvegarde des différentes valeurs,
3. la base de donnée doit être accessible via le web.

1.2 Composante matérielle

En se basant sur les recommendations de la section 1.1, la selection des équippements peut être raffinée. Comment peut effectuer le suivi de l'activité cardiaque ? l'un des capteur permettant une telle opération est le capteur AD8232. La section suivante donne plus de détails.

1.2.1 Le capteur AD8232

Le module AD8232 assure la surveillance des impulsions cardiaque. Le kit apporte en plus 3 électrodes à placer sur le corp du patient. Ce module, dont le datasheet est disponible sur ce lien (voir Annexe A), est caractérisé principalement par :

— un filtre passe-haut à deux pôles pour éliminer les artefacts de mouvement et le potentiel de demi-cellule d'électrode
— un amplificateur opérationnel sans contrainte pour construire un filtre passe-bas à trois pôles éliminant les bruits suplémentaires
— température nominale de $[0;70]$ et de travail $[-40;85]$
— alimentation faible de 3,3V
— il est conçu pour extraire, amplifier et filtrer les petits signaux bipotentiels en présence de conditions bruyantes.

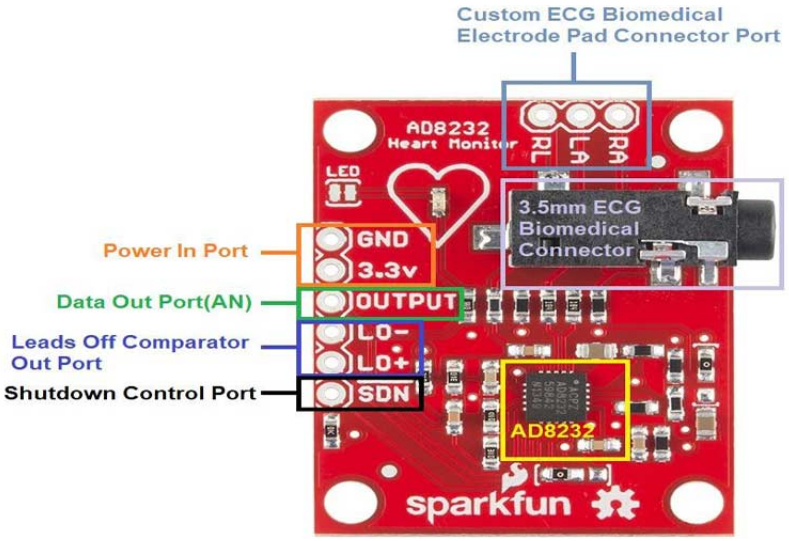

FIGURE 1.1 – Description du capteur AD8232

Le tableau 1.1 donne les pins du capteur

PIN	Description
`GND`	la masse
`3.3V`	alimentation
`Output` (ADC)	la sortie traitée du signal
`LO-`	leads off detection mode. `LO-` est `HIGH` si l'électrode rouge est déconnectée, et `LOW` sinon
`LO+`	leads off detection mode. `LO+` est `HIGH` si l'électrode jaune est déconnectée, et `LOW` sinon
`SDN`	Shutdown Control Input. Si cette pin est `LOW`, le capteur active le mode de faible consommation

TABLE 1.1 – AD8232 PINOUT

1.2.2 Carte de développement

La section 1.2.1 a détaillé les carectéristiques techniques du capteur. Maintenant, une foi le capteur délivre l'activité électrique via sa pin `Output`, vers quelle carte de développement cette information sera finalement acheminée ?

Pour répondre à cette question, il est recommander de revoir les recommendations citées dans la section 1.1. En effet, l'activité cardiaque doit être envoyée vers le WEB. Ce qui impose que la carte doit pouvoir se connecter à INTERNET. Une recherche comparative entre les différents boards éligibles

tels que Arduino (avec ses variantes), STM32, ESP32 . . .a été établie. Les critères de sélection retenus sont la connectivité, le prix et la disponibilité. Finalement, la carte ESP32 est celle qui répond mieux aux différentes contraintes. Elle offre plusieurs types de connectivités (WIFI, Bluetooth) avec un prix minimal.

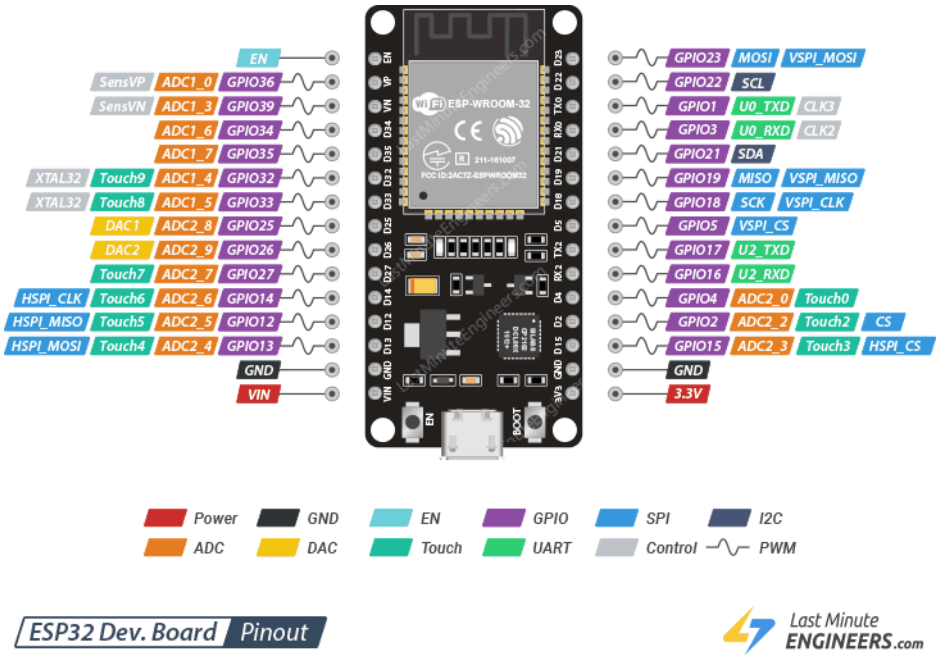

FIGURE 1.2 – Description de la carte de développement ESP32

L'ESP32 est un microcontroleur dont le constructeur est *Espressif*. Ce constructeur a développé tout un système temps réel (FreeRTOS) pour ce microcontroleur. C'est pourquoi il est bien adapté aux applications temps réel et IoT. Il est caractérisé principalement par :

- CPU : Xtensa double-cœur (ou simple-cœur), microprocesseur LX 32 bits, fonctionnant à 160 ou 240 MHz et fournissant jusqu'à 600 DMIPS, avec un coprocesseur ultra basse consommation (ULP)
- Mémoire : 520 KiO SRAM
- Connectivité sans-fil : Wi-Fi : 802.11 b/g/n; Bluetooth : v 4.2 BR/EDR et BLE jusqu'à v 5.0 et v 5.1
- 10 × capteurs de touché
- 4 × SPI
- 2 × interfaces I²S
- 2 × interfaces I²C
- 3 × UART
- interface MAC Ethernet avec DMA dédié et support du protocole de temps précis IEEE 1588

— Bus de données CAN 2.0
— Moteur PWM

1.2.3 Assemblage

La connection du capteur AD8232 avec la carte ESP32 doit suivre le schéma suivant :

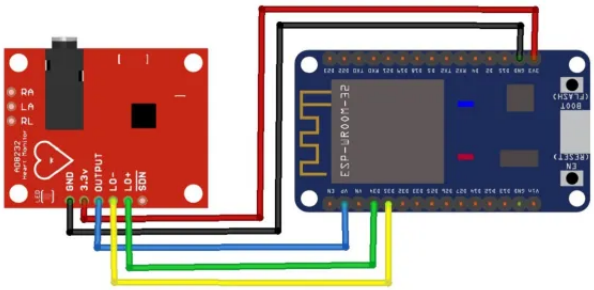

FIGURE 1.3 – Interfaçage du capteur AD8232 avec la carte de développement ESP32

Pour que le système soit en marche, il faut ajouter :

1. une connexion INTERNET : la carte ESP32 peut se connecter à un réseau WIFI. Mais un tel réseau doit assurer une connexion à INTERNET et n'est pas un simple réseau local par exemple.
2. un serveur de données : il faut avoir un accès à un serveur dans le WEB afin d'y enregistrer les informations récupérées du capteur. Ainsi, une inscription dans une plateforme dédiée peut donner ce type d'accès.

Jusqu'à présent, la partie hardware du projet a été investiguée. En se basant sur cette architecture, quels sont les composants software à ajouter permettant le bon fonctionnement du système ?

1.3 Composante logicielle

La couche logicielle est composée de deux parties : un programme qui doit être dépolyé dans la carte ESP32 permettant la collecte et la communication des signaux ECG, et d'une configuration au niveau du cloud ubidots permettant la savegarde et l'exploitation des données envoyées.

1.3.1 Programmation de l'ESP32

Le programme détaillé est présent dans l'annexe B.1. Nous pouvons énumérer les principales étapes suivantes :

1. importation des bibliothèques nécessaires. Il faut souligner une installation au préalable des deux bibliothèques `PubSubClient` et `NTPClient`. Comme nous utilisons PlatformIO pour programmer le circuit, il faut mettre à jour le ficher plateforme.ini.
2. donner les paramètres d'accès au réseau WIFI, au serveur MQTT et à la plateforme ubidots.
3. maintenant, en boucle, collecter 4 lectures avec une période de 150ms et les communiquer à la plateforme ubidots.

1.3.2 Configuration de Ubidots

Avant de se lancer dans la programmation, la Configuration de la plateforme Ubidots est nécessaire afin de préparer la hôte pour les données envoyées par le capteur.

Après une inscription, la première étape consiste à créer une nouveau composant comme le montre la figure 1.4.

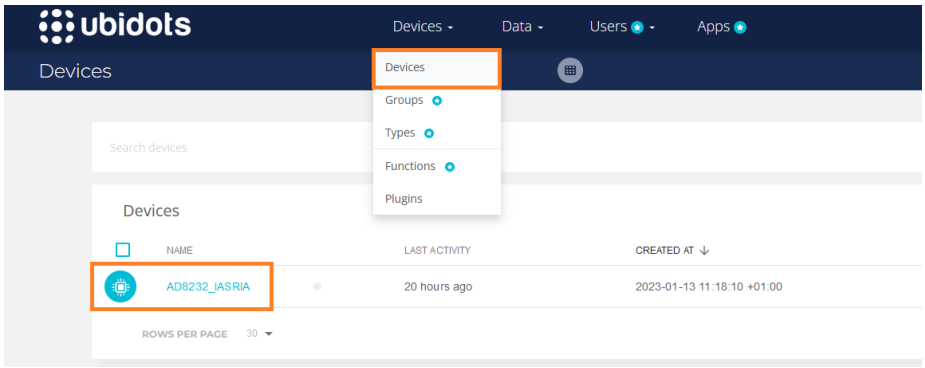

FIGURE 1.4 – Description du capteur AD8232

La plateforme offre une large variante de composants comme la montre la figure 1.5

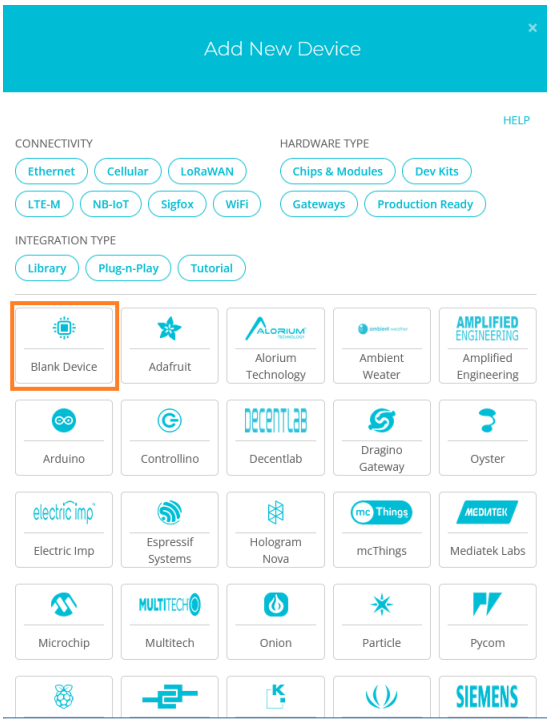

FIGURE 1.5 – Description du capteur AD8232

Maintenant, il faut attribuer une variable à ce composant comme le montre la figure 1.6

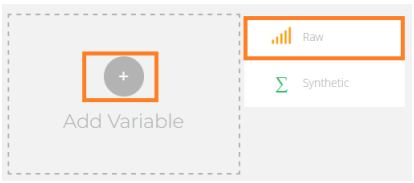

FIGURE 1.6 – Description du capteur AD8232

Finalement, la variable `ecg_val` est créé comme le montre la figure 1.7. Cette variable est utilisée par le programme afin de communiquer les données au capteur à la plateforme (voir l'appendix B.1).

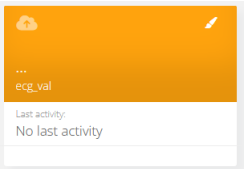

FIGURE 1.7 – Description du capteur AD8232

D'un autre coté, il faut ajouter un nouveau dashboard pour visualiser les données, comme le montre la figure 1.8

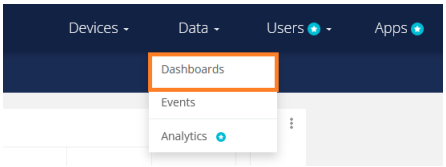

FIGURE 1.8 – Description du capteur AD8232

Une grande variété de widgets est offerte afin de s'adapter aux différents besoins de l'utilisateur. Comme nous retraçons l'activité électrique cardiaque, nous voulons voir son évolution sous forme d'une courbe comme le montre la figure 1.9

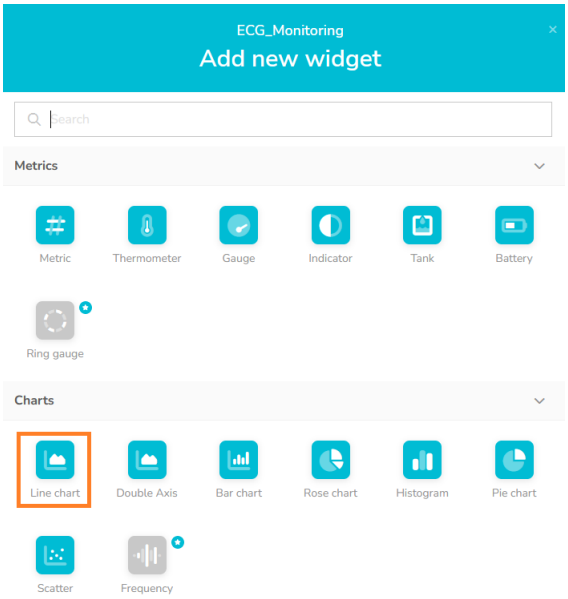

FIGURE 1.9 – Description du capteur AD8232

Conclusion générale

Nous avons essayé, dans le cadre de ce projet, de s'adresser à problème sérieux à savoir le suivi de l'activité cardiologique humaine. Une contrainte imposée de plus en plus dans un environnement accidenté nécessitant la surveillance vigilante de l'état de santé.

Ainsi, nous proposons une approche basée sur IoT. Ce choix se justifie par le besoin de pouvoir consulter et exploiter les données collectées en temps réel. Nous avons utilisé le capteur AD8232 pour la détection de l'activité cardiaque. D'un autre coté la carte ESP32 offre l'avantage de se connecter directement au réseau (sans avoir ajouter un autre composant) avec prix raisonnable. Nous avons choisi la plateforme ubidots pour l'enregistrement des données.

Finalement, nous avons réussi à enregistrer les données dans le cloud. D'autres traitements peuvent s'appliquer par la suite telle que la détection d'anomalies cardiologiques. En effet, ajouter une couche intelligente permettra la détection au préalable des problèmes cardiologiques.

Annexe A

Datasheet du capteur AD8232

Single-Lead, Heart Rate Monitor Front End

Data Sheet **AD8232**

FEATURES

- Fully integrated single-lead ECG front end
- Low supply current: 170 µA (typical)
- Common-mode rejection ratio: 80 dB (dc to 60 Hz)
- Two or three electrode configurations
- High signal gain (G = 100) with dc blocking capabilities
- 2-pole adjustable high-pass filter
- Accepts up to ±300 mV of half cell potential
- Fast restore feature improves filter settling
- Uncommitted op amp
- 3-pole adjustable low-pass filter with adjustable gain
- Leads off detection: ac or dc options
- Integrated right leg drive (RLD) amplifier
- Single-supply operation: 2.0 V to 3.5 V
- Integrated reference buffer generates virtual ground
- Rail-to-rail output
- Internal RFI filter
- 8 kV HBM ESD rating
- Shutdown pin
- 20-lead 4 mm × 4 mm LFCSP package

APPLICATIONS

- Fitness and activity heart rate monitors
- Portable ECG
- Remote health monitors
- Gaming peripherals
- Biopotential signal acquisition

FUNCTIONAL BLOCK DIAGRAM

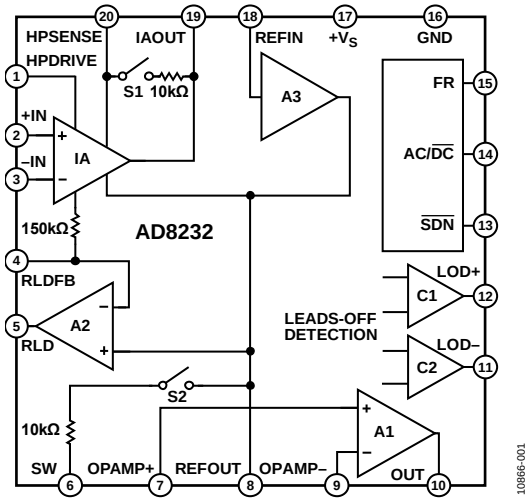

Figure 1.

GENERAL DESCRIPTION

The AD8232 is an integrated signal conditioning block for ECG and other biopotential measurement applications. It is designed to extract, amplify, and filter small biopotential signals in the presence of noisy conditions, such as those created by motion or remote electrode placement. This design allows for an ultralow power analog-to-digital converter (ADC) or an embedded microcontroller to acquire the output signal easily.

The AD8232 can implement a two-pole high-pass filter for eliminating motion artifacts and the electrode half-cell potential. This filter is tightly coupled with the instrumentation architecture of the amplifier to allow both large gain and high-pass filtering in a single stage, thereby saving space and cost.

An uncommitted operational amplifier enables the AD8232 to create a three-pole low-pass filter to remove additional noise. The user can select the frequency cutoff of all filters to suit different types of applications.

To improve common-mode rejection of the line frequencies in the system and other undesired interferences, the AD8232 includes an amplifier for driven lead applications, such as right leg drive (RLD).

The AD8232 includes a fast restore function that reduces the duration of otherwise long settling tails of the high-pass filters. After an abrupt signal change that rails the amplifier (such as a leads off condition), the AD8232 automatically adjusts to a higher filter cutoff. This feature allows the AD8232 to recover quickly, and therefore, to take valid measurements soon after connecting the electrodes to the subject.

The AD8232 is available in a 4 mm × 4 mm, 20-lead LFCSP package. Performance is specified from 0°C to 70°C and is operational from −40°C to +85°C.

Rev. A Document Feedback

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700
Technical Support www.analog.com

SPECIFICATIONS

V_S = 3 V, V_{REF} = 1.5 V, V_{CM} = 1.5 V, T_A = 25°C, FR=low, SDN=high, AC/$\overline{DC}$ = low, unless otherwise noted.

Table 1.

Parameter	Symbol	Test Conditions/Comments	Min	Typ	Max	Unit
INSTRUMENTATION AMPLIFIER						
Common-Mode Rejection Ratio, DC to 60 Hz	CMRR	V_{CM} = 0.35 V to 2.85 V, V_{DIFF} = 0 V	80	86		dB
		V_{CM} = 0.35 V to 2.85 V, V_{DIFF} = ±0.3 V		80		dB
Power Supply Rejection Ratio	PSRR	V_S = 2.0 V to 3.5 V	76	90		dB
Offset Voltage (RTI)	V_{OS}					
Instrumentation Amplifier Inputs				3	8	mV
DC Blocking Input[1]				5	50	µV
Average Offset Drift						
Instrumentation Amplifier Inputs				10		µV/°C
DC Blocking Input[1]				0.05		µV/°C
Input Bias Current	I_B			50	200	pA
		T_A = 0°C to 70°C		1		nA
Input Offset Current	I_{OS}			25	100	pA
		T_A = 0°C to 70°C		1		nA
Input Impedance						
Differential				10||7.5		GΩ||pF
Common Mode				5||15		GΩ||pF
Input Voltage Noise (RTI)						
Spectral Noise Density		f = 1 kHz		100		nV/√Hz
Peak-to-Peak Voltage Noise		f = 0.1 Hz to 10 Hz		12		µV p-p
		f = 0.5 Hz to 40 Hz		14		µV p-p
Input Voltage Range		T_A = 0°C to 70°C	0.2		$+V_S$	V
DC Differential Input Range	V_{DIFF}		−300		+300	mV
Output						
Output Swing		R_L = 50 kΩ	0.1		$+V_S$ − 0.1	V
Short-Circuit Current	I_{OUT}			6.3		mA
Gain	A_V			100		V/V
Gain Error		V_{DIFF} = 0 V		0.4		%
		V_{DIFF} = −300 mV to +300 mV		1	3.5	%
Average Gain Drift		T_A = 0°C to 70°C		12		ppm/°C
Bandwidth	BW			2		kHz
RFI Filter Cutoff (Each Input)				1		MHz
OPERATIONAL AMPLIFIER (A1)						
Offset Voltage	V_{OS}			1	5	mV
Average TC		T_A = 0°C to 70°C		5		µV/°C
Input Bias Current	I_B			100		pA
		T_A = 0°C to 70°C		1		nA
Input Offset Current	I_{OS}			100		pA
		T_A = 0°C to 70°C		1		nA
Input Voltage Range			0.1		$+V_S$ − 0.1	V
Common-Mode Rejection Ratio	CMRR	V_{CM} = 0.5 V to 2.5 V		100		dB
Power Supply Rejection Ratio	PSRR			100		dB
Large Signal Voltage Gain	A_{VO}			110		dB
Output Voltage Range		R_L = 50 kΩ	0.1		$+V_S$ − 0.1	V
Short-Circuit Current Limit	I_{OUT}			12		mA
Gain Bandwidth Product	GBP			100		kHz
Slew Rate	SR			0.02		V/µs
Voltage Noise Density (RTI)	e_n	f = 1 kHz		60		nV/√Hz
Peak-to-Peak Voltage Noise (RTI)	$e_{n\ p\text{-}p}$	f = 0.1 Hz to 10 Hz		6		µV p-p
		f = 0.5 Hz to 40 Hz		8		µV p-p

Parameter	Symbol	Test Conditions/Comments	Min	Typ	Max	Unit
RIGHT LEG DRIVE AMPLIFIER (A2)						
Output Swing		R_L = 50 kΩ	0.1		$+V_S$ − 0.1	V
Short-Circuit Current	I_{OUT}			11		mA
Integrator Input Resistor			120	150	180	kΩ
Gain Bandwidth Product	GDP			100		kHz
REFERENCE BUFFER (A3)						
Offset Error	V_{OS}	R_L > 50 kΩ		1		mV
Input Bias Current	I_B			100		pA
Short-Circuit Current Limit	I_{OUT}			12		mA
Voltage Range		R_L = 50 kΩ	0.1		$+V_S$ − 0.7	V
DC LEADS OFF COMPARATORS						
Threshold Voltage				$+V_S$ − 0.5		V
Hysteresis				60		mV
Propagation Delay				0.5		µs
AC LEADS OFF DETECTOR						
Square Wave Frequency	F_{AC}		50	100	175	kHz
Square Wave Amplitude	I_{AC}			200		nA p-p
Impedance Threshold		Between +IN and −IN	10	20		MΩ
Detection Delay				110		µs
FAST RESTORE CIRCUIT						
Switches		S1 and S2				
On Resistance	R_{ON}		8	10	12	kΩ
Off Leakage				100		pA
Window Comparator						
Threshold Voltage		From either rail		50		mV
Propagation Delay				2		µs
Switch Timing Characteristics						
Feedback Recovery Switch On Time	t_{SW1}			110		ms
Filter Recovery Switch On Time	t_{SW2}			55		ms
Fast Restore Reset	t_{RST}			2		µs
LOGIC INTERFACE						
Input Characteristics						
Input Voltage (AC/$\overline{DC}$ and FR)						
Low	V_{IL}			1.24		V
High	V_{IH}			1.35		V
Input Voltage ($\overline{SDN}$)						
Low	V_{IL}			2.1		V
High	V_{IH}			0.5		V
Output Characteristics		LOD+ and LOD− terminals				
Output Voltage						
Low	V_{OL}			0.05		V
High	V_{OH}			2.95		V
SYSTEM SPECIFICATIONS						
Quiescent Supply Current				170	230	µA
		T_A = 0°C to 70°C		210		µA
Shutdown Current				40	500	nA
		T_A = 0°C to 70°C		100		nA
Supply Range			2.0		3.5	V
Specified Temperature Range			0		70	°C
Operational Temperature Range			−40		+85	°C

[1] Offset referred to the input of the instrumentation amplifier inputs. See the Input Referred Offsets section for additional information.

ABSOLUTE MAXIMUM RATINGS

Table 2.

Parameter	Rating
Supply Voltage	3.6 V
Output Short-Circuit Current Duration	Indefinite
Maximum Voltage, Any Terminal[1]	$+V_S$ + 0.3 V
Minimum Voltage, Any Terminal[1]	−0.3 V
Storage Temperature Range	−65°C to +125°C
Operating Temperature Range	−40°C to +85°C
Maximum Junction Temperature	140°C
θ_{JA} Thermal Impedance[2]	48°C/W
θ_{JC} Thermal Impedance	4.4°C/W
ESD Rating	
Human Body Model (HBM)	8 kV
Charged Device Model (FICDM)	1.25 kV
Machine Model (MM)	200 V

[1] This level or the maximum specified supply voltage, whichever is the lesser, indicates the superior voltage limit for any terminal. If input voltages beyond the specified minimum or maximum voltages are expected, place resistors in series with the inputs to limit the current to less than 5 mA.

[2] θ_{JA} is specified for a device in free air on a 4-layer JEDEC board.

Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ESD CAUTION

ESD (electrostatic discharge) sensitive device. Charged devices and circuit boards can discharge without detection. Although this product features patented or proprietary protection circuitry, damage may occur on devices subjected to high energy ESD. Therefore, proper ESD precautions should be taken to avoid performance degradation or loss of functionality.

Annexe B

Programme

B.1 Code source

```
  /**
* @file main.cpp
* @author N. AZZOUZ, A. MELLAH, R. ALOUI
* @brief
* @version 0.1
* @date 2023-01-14
*
* @copyright Copyright (c) 2023
*
*/
#include <Arduino.h>

#include <WiFi.h>
#include <WiFiUdp.h>  // Utilisation du protocole UDP (pour les
↪  applications temps réel)
#include <PubSubClient.h>
#include <NTPClient.h>


#define WIFISSID "Redmi Note 11" // Enter WifiSSID here
#define PASSWORD "isetbeja" // Enter password here
#define TOKEN "BBFF-Hz6tdZJl9805kiV1KRrxKrXaEjIV24" // Ubidots' TOKEN
#define MQTT_CLIENT_NAME "BBFF-Hz6tdZJl9805kiV1KRrxKrXaEjIV24" // MQTT
↪  client Name
#define VARIABLE_LABEL "ecg_val" // ubidots variable label
#define DEVICE_LABEL "AD8232_IASRIA" // ubidots device label
#define T_SAMPLING 150 // Period of sampling (ms)


#define SENSORPIN A0 // Set the A0 as SENSORPIN

char mqttBroker[]  = "industrial.api.ubidots.com";
char payload[10000];
char topic[150];
```

```
33 // Space to store values to send
34 char str_sensor[10];
35 char str_millis[20];
36 double epochseconds = 0;
37 double epochmilliseconds = 0;
38 double current_millis = 0;
39 double current_millis_at_sensordata = 0;
40 double timestampp = 0;
41 int j = 0;
42 /****************************************
43  Auxiliar Functions
44 ****************************************/
45 WiFiClient ubidots;
46 PubSubClient client(ubidots);
47 WiFiUDP ntpUDP;
48 NTPClient timeClient(ntpUDP, "pool.ntp.org");
49
50 void callback(char* topic, byte* payload, unsigned int length) {
51 char p[length + 1];
52 memcpy(p, payload, length);
53 p[length] = NULL;
54 Serial.write(payload, length);
55 Serial.println(topic);
56 }
57
58 void reconnect() {
59 // Loop until we're reconnected
60 while (!client.connected()) {
61   Serial.println("Attempting MQTT connection...");
62
63   // Attemp to connect
64   if (client.connect(MQTT_CLIENT_NAME, TOKEN, "")) {
65     Serial.println("Connected");
66   } else {
67     Serial.print("Failed, rc=");
68     Serial.print(client.state());
69     Serial.println(" try again in 2 seconds");
70     // Wait 2 seconds before retrying
71     delay(2000);
72   }
73 }
74 }
75
76 /****************************************
77  setup function
78 ****************************************/
79 void setup() {
80 Serial.begin(115200);
81 WiFi.begin(WIFISSID, PASSWORD);
82 // Assign the pin as INPUT
83 pinMode(SENSORPIN, INPUT);
84
85 Serial.println();
86 Serial.print("Waiting for WiFi...");
87
```

```
88  while (WiFi.status() != WL_CONNECTED) { // Try to connect to network
89    Serial.print(".");
90    delay(500);
91  }
92
93  Serial.println("");
94  Serial.println("WiFi Connected");
95  Serial.println("IP address: ");
96  Serial.println(WiFi.localIP());
97  timeClient.begin();
98  client.setServer(mqttBroker, 1883); // Try to connect to MQTT server
99  client.setCallback(callback);
100 timeClient.update();
101 epochseconds = timeClient.getEpochTime();
102 epochmilliseconds = epochseconds * 1000;
103 Serial.print("epochmilliseconds=");
104 Serial.println(epochmilliseconds);
105 current_millis = millis();
106 Serial.print("current_millis=");
107 Serial.println(current_millis);
108 }
109
110 /****************************************
111  setup function
112 ****************************************/
113 void loop() {
114 if (!client.connected()) {
115   reconnect();
116   j = 0;
117 }
118
119 j = j + 1;  // Messages sent
120 Serial.print("j=");
121 Serial.println(j);
122 sprintf(topic, "%s%s", "/v1.6/devices/", DEVICE_LABEL);
123 sprintf(payload, "%s", ""); // Cleans the payload
124 sprintf(payload, "{\"%s\": [", VARIABLE_LABEL); // Adds the variable
    ↪  label
125 for (int i = 1; i <= 3; i++)
126 {
127   float sensor = analogRead(SENSORPIN); // Read from sensor
128   dtostrf(sensor, 4, 2, str_sensor);
129   sprintf(payload, "%s{\"value\":", payload); // Adds the value
130   sprintf(payload, "%s %s,", payload, str_sensor); // Adds the value
131   current_millis_at_sensordata = millis();
132   timestampp = epochmilliseconds + (current_millis_at_sensordata -
    ↪  current_millis);
133   dtostrf(timestampp, 10, 0, str_millis);
134   sprintf(payload, "%s \"timestamp\": %s},", payload, str_millis); //
    ↪  Adds the value
135   delay(T_SAMPLING);
136 }
137
138 float sensor = analogRead(SENSORPIN); // Read from sensor
139 dtostrf(sensor, 4, 2, str_sensor);
```

```
140 current_millis_at_sensordata = millis();
141 timestampp = epochmilliseconds + (current_millis_at_sensordata -
    ↪ current_millis);
142 dtostrf(timestampp, 10, 0, str_millis);
143 sprintf(payload, "%s{\"value\":%s, \"timestamp\": %s}]}", payload,
    ↪ str_sensor, str_millis);
144 Serial.println("Publishing data to Ubidots Cloud");
145 client.publish(topic, payload); // Publish data to ubidots.com
146 Serial.println(payload);
147 delay(T_SAMPLING);
148 }
149
```

B.2 Output

```
  Waiting for WiFi.........
WiFi Connected
IP address:
192.168.1.6
epochmilliseconds=ovf
current_millis=3362.00
Attempting MQTT connection...
Connected
j=1
Publishing data to Ubidots Cloud
{"ecg_val": [{"value": 4095.00, "timestamp":
    1673654680417},{"value": 4095.00, "timestamp":
    1673654680567},{"value": 4095.00, "timestamp":
    1673654680717},{"value":4095.00, "timestamp": 1673654680867}]}
j=2
Publishing data to Ubidots Cloud
{"ecg_val": [{"value": 4095.00, "timestamp":
    1673654680878},{"value": 4095.00, "timestamp":
    1673654681029},{"value": 4095.00, "timestamp":
    1673654681179},{"value":4095.00, "timestamp": 1673654681329}]}
j=3
Publishing data to Ubidots Cloud
{"ecg_val": [{"value": 4095.00, "timestamp":
    1673654681339},{"value": 4095.00, "timestamp":
    1673654681489},{"value": 4095.00, "timestamp":
    1673654681639},{"value":4095.00, "timestamp": 1673654681789}]}
j=4
Publishing data to Ubidots Cloud
{"ecg_val": [{"value": 4095.00, "timestamp":
    1673654681800},{"value": 4095.00, "timestamp":
    1673654681950},{"value": 4095.00, "timestamp":
```

```
    1673654682100},{"value":4095.00, "timestamp": 1673654682250}]}
j=5
Publishing data to Ubidots Cloud
{"ecg_val": [{"value": 4095.00, "timestamp":
    1673654682260},{"value": 4095.00, "timestamp":
    1673654682410},{"value": 4095.00, "timestamp":
    1673654682560},{"value":4095.00, "timestamp": 1673654682710}]}
j=6
Publishing data to Ubidots Cloud
{"ecg_val": [{"value": 4095.00, "timestamp":
    1673654682721},{"value": 4095.00, "timestamp":
    1673654682871},{"value": 4095.00, "timestamp":
    1673654683021},{"value":4095.00, "timestamp": 1673654683171}]}
j=7
Publishing data to Ubidots Cloud
{"ecg_val": [{"value": 4095.00, "timestamp":
    1673654683181},{"value": 4095.00, "timestamp":
    1673654683331},{"value": 4095.00, "timestamp":
    1673654683481},{"value":4095.00, "timestamp": 1673654683631}]}
j=8
Publishing data to Ubidots Cloud
{"ecg_val": [{"value": 4095.00, "timestamp":
    1673654683642},{"value": 4095.00, "timestamp":
    1673654683793},{"value": 4095.00, "timestamp":
    1673654683943},{"value":4095.00, "timestamp": 1673654684093}]}
j=9
Publishing data to Ubidots Cloud
{"ecg_val": [{"value": 4095.00, "timestamp":
    1673654684103},{"value": 4095.00, "timestamp":
    1673654684253},{"value": 4095.00, "timestamp":
    1673654684403},{"value":4095.00, "timestamp": 1673654684553}]}
j=10
Publishing data to Ubidots Cloud
{"ecg_val": [{"value": 4095.00, "timestamp":
    1673654684564},{"value": 4095.00, "timestamp":
    1673654684715},{"value": 4095.00, "timestamp":
    1673654684865},{"value":4095.00, "timestamp": 1673654685015}]}
j=11
Publishing data to Ubidots Cloud
{"ecg_val": [{"value": 4095.00, "timestamp":
    1673654685025},{"value": 4095.00, "timestamp":
    1673654685176},{"value": 4095.00, "timestamp":
    1673654685326},{"value":4095.00, "timestamp": 1673654685476}]}
j=12
Publishing data to Ubidots Cloud
{"ecg_val": [{"value": 4095.00, "timestamp":
    1673654685487},{"value": 4095.00, "timestamp":
    1673654685638},{"value": 4095.00, "timestamp":
```

```
1673654685788},{"value":4095.00, "timestamp": 1673654685938}]}
```

Bibliographie

[1] Jamin Casselman, Nicholas Onopa, and Lara Khansa. Wearable healthcare : Lessons from the past and a peek into the future. *Telematics and Informatics*, 34(7) :1011–1023, 2017.

[2] Lei Ma, Hidemichi Kiyomatsu, Keiichi Nakagawa, Junchen Wang, Etsuko Kobayashi, and Ichiro Sakuma. Accurate vessel segmentation in ultrasound images using a local-phase-based snake. *Biomedical Signal Processing and Control*, 43 :236–243, 2018.

[3] Özal Yıldırım, Paweł Pławiak, Ru-San Tan, and U. Rajendra Acharya. Arrhythmia detection using deep convolutional neural network with long duration ecg signals. *Computers in Biology and Medicine*, 102 :411–420, 2018.