Heart Disease Detection in STM32 microcontroller

https://github.com/djzenma/ECG

Mazen Amr Gamaleldin Ahmed

900161021

Motivation

- Embedded Systems are found everywhere, and you might even be interacting with some of them without even noticing that it is one. This is due to their large field of applications such as cars (ABS systems and others), Point-Of-Sale systems, anti-focus cameras, avionic systems, etc.... and this is just to name a few.
- Embedded Systems are usually cheap as they are often part of a bigger system that should be cheap. This implies that embedded systems are accessible for anyone.
  Recently, embedded systems begun to support the implementation of Artificial Intelligence (we will discuss this later). Some of these microcontrollers are, of course, the STM32 family.

Motivation

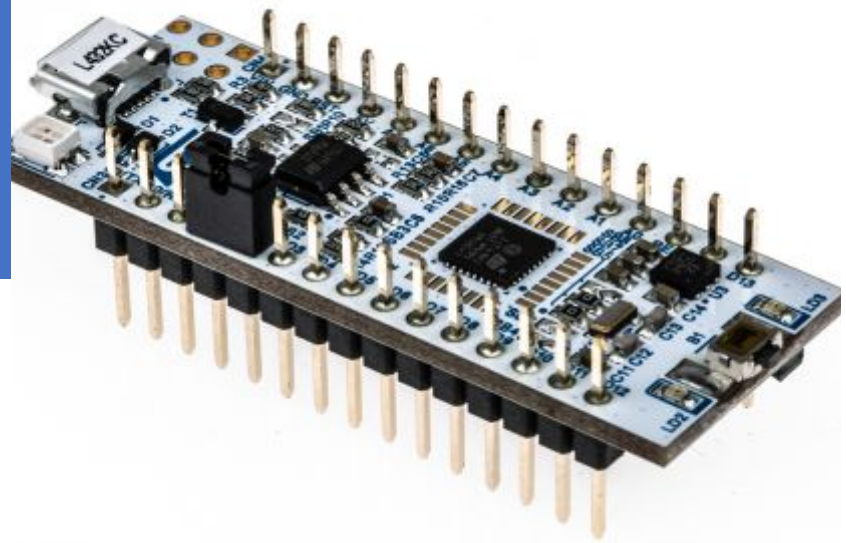
- Artificial Intelligence (AI) is called “The New Electricity” by some scientists, such as Andrew Ng. This is because AI has found its way in every domain of our lives, starting from our alarms to medicine and diseases detection. Indeed, AI has proven its excellence in medicine and helped in many developed countries to speed up diagnosis and increase the number of possible examinations per day.
- If we focused in heart diseases, AI is able to diagnose heart problems as good as a human doctor but in just a few seconds. If we let a human do this same task, it will require him about 13 minutes per case, which is obviously a huge speed up.

Motivation

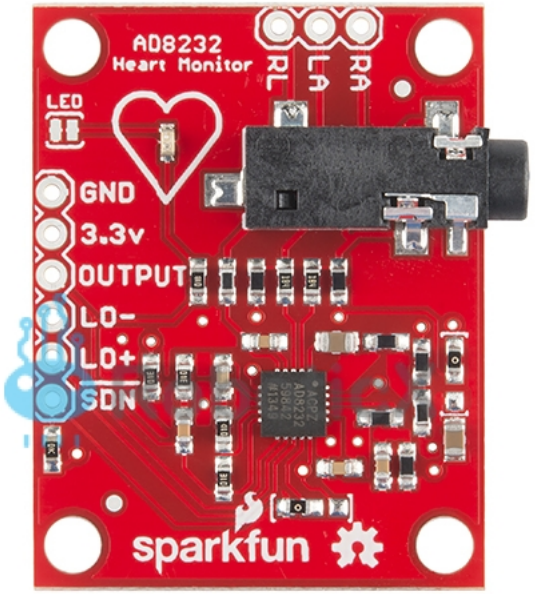
If we combined the power of both worlds, i.e. the diagnosis and speed power of AI and the computation power and accessibility of Embedded Systems, we will obtain our beautiful project:

Heart Disease Detection in STM32 microcontroller.

Components Needed (Hardware)



- A Microcontroller, I will use the Nucleo STM32L432. The Nucleo has built-in ADC and UART modules, if your MCU doesn't contain any of these modules you will have to acquire them and connect them to your MCU.
- A Heart Monitor, I will use the Spark Fun AD8232
- Breadboard
- Some jumpers



Components Needed (Software)

- Tensorflow Lite Support
- An IDE, I will use Keil uVision 5
- STM32CubeMX, this is not required but it facilitates the implementation a lot

Architecture

Firm Deadline

Our application's deadlines are categorized as Firm deadlines. In other words, it is okay if a couple of deadlines are missed as this will not halt the performance of our system. The deadline in our case is to successfully collect 125 samples of ECG signal and classify the disease state of the patient.

Architecture

ADC

I used the built-in ADC in my Nucleo and configured it as follows:

- Port: PA0
- ADC1 channel IN5 Single-Ended mode
- Resolution = 12 bits

Architecture

Timer

- TIM15
- Its clock = 1KHz
  - Original Clock = 10MHz
  - Prescaler = 9,999
  - Resultant clock = 10MHz / 9,999+1 = 1KHz
- Auto-Reload Register = 8
- **Sampling rate** = 1KHz/8 = 125Hz

Architecture

GPIO

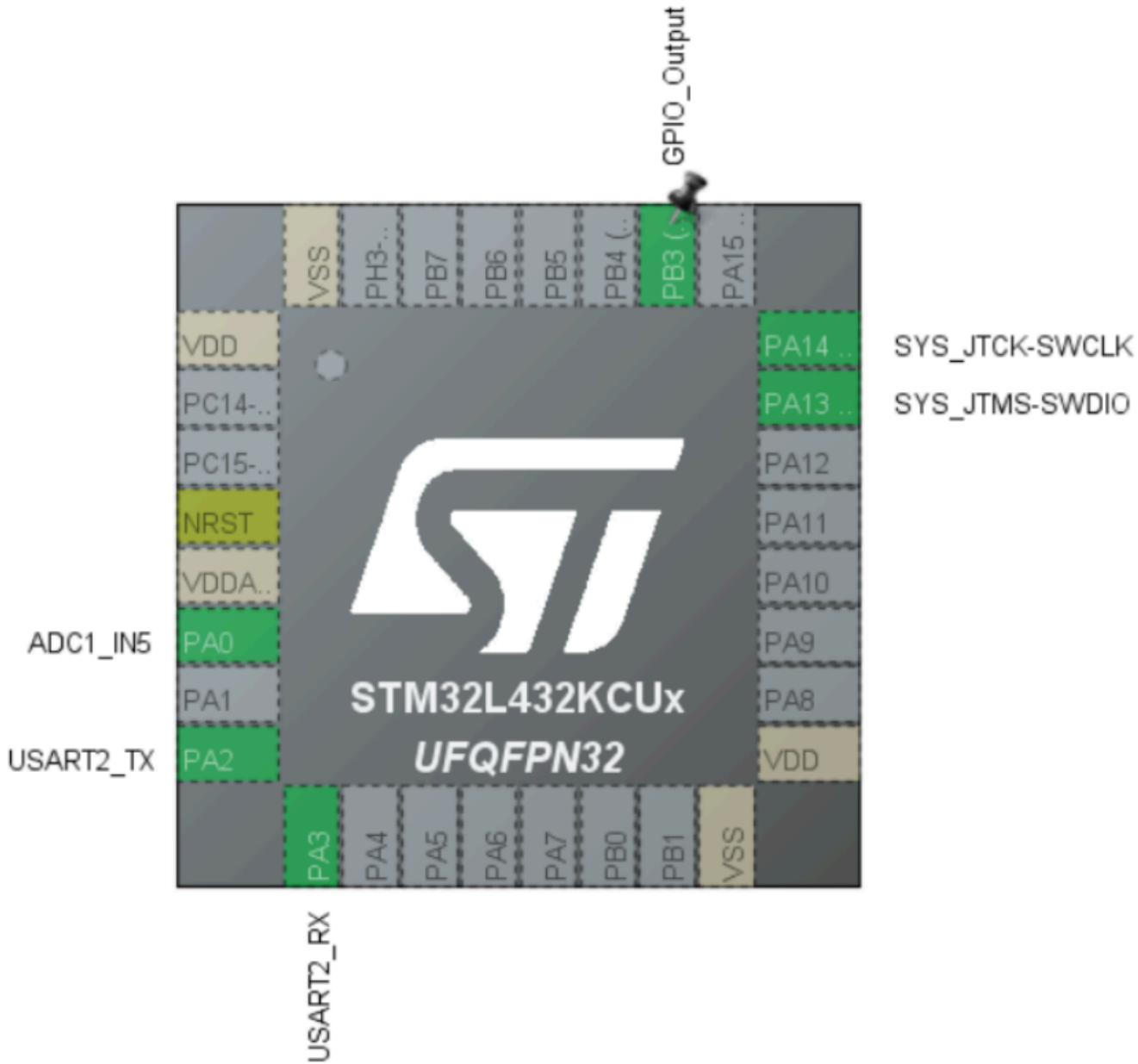
- 2 input Pins: PA9, PA10
- 1 output Pin: PB3

UART

- Ports used: PA2 of Transmission and PA3 for Receiving
- Baud Rate = 9600 bits/s

Internal Clock

- SYSCLK = 80 MHz

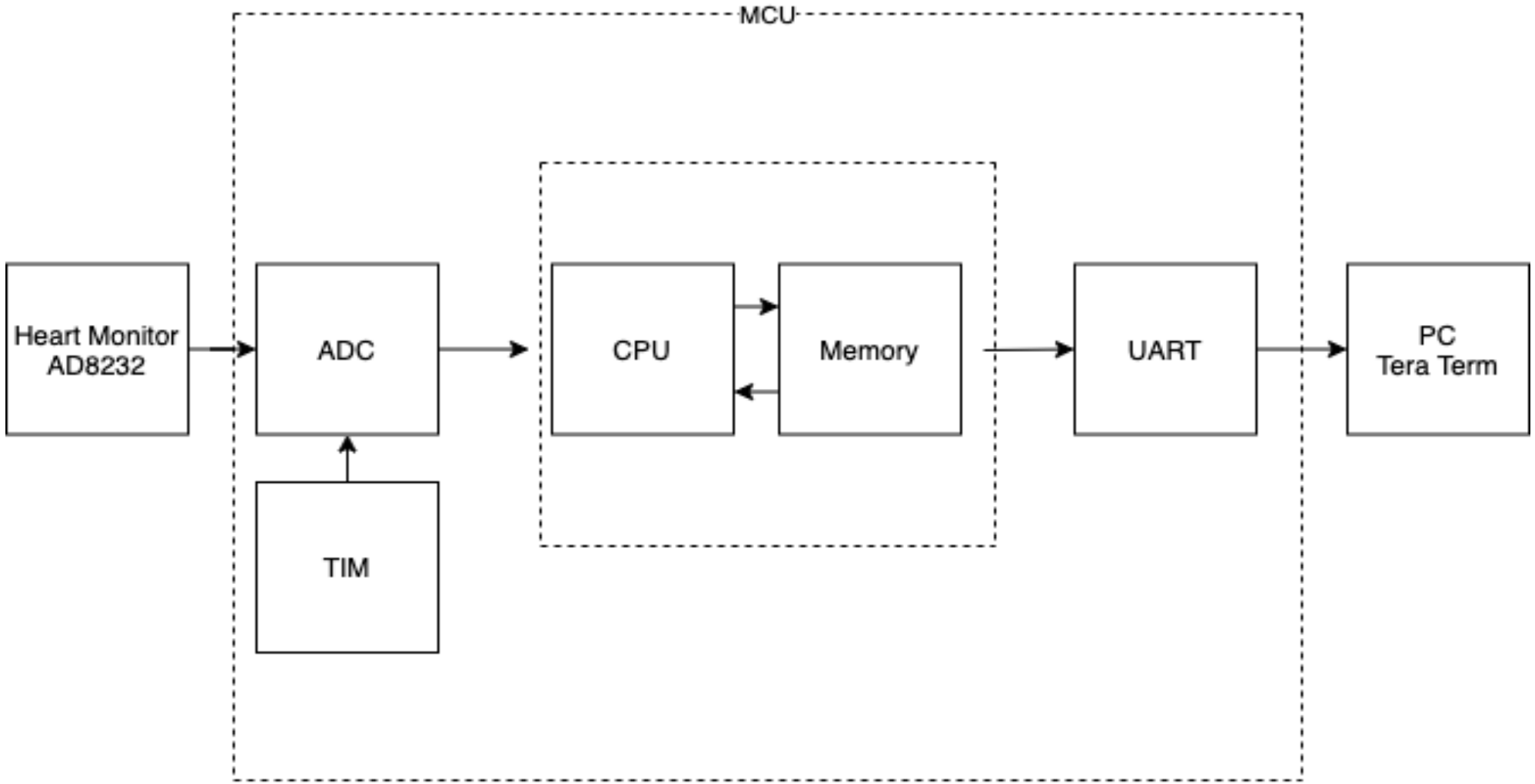

GPIO_Output
VSS
PH3-..
PB7
PB6
PB5
PB4 (..
PB3 (..
PA15 ..
VDD
PC14-..
PC15-..
NRST
VDDA..
PA0
PA1
PA2
PA14 ..
PA13 ..
PA12
PA11
PA10
PA9
PA8
VDD
STM32L432KCUx
UFQFPN32
ADC1_IN5
USART2_TX
SYS_JTCK-SWCLK
SYS_JTMS-SWDIO
PA3
PA4
PA5
PA6
PA7
PB0
PB1
VSS
USART2_RX

Architecture

Connections

Heart Monitor	Nucleo
GND	GND
3.3V	3.3V
OUTPUT	PA0 (ADC1 CH1)
Lo-	PA9 (GPIO Input)
Lo+	PA10 (GPIO Input)

Architecture Overview



Architecture Overview

- The Heart Monitor sends the ECG signal as an Analog signal through its OUTPUT pin.
- The ADC samples it and converts it to a digital signal every 8ms.
- This timing is done using a Timer that raises an interrupt every time the timer reaches the value set in the AR register, which is 8. This allows us to achieve 1000KHz/8 = 125Samples/Sec.
- This digital sample is processed, saved in the memory until 125 signals are formed then fed to the ML model to predict the patient's disease status.
- This patient's status is then sent through the UART to the PC Tera Term as a string specifying to which class does the patient belong.
- Finally, the 125 signals are flushed and this cycle repeats.

Authors Signal Preprocessing*

1) Splitting the continuous ECG signal to 10s windows and select a 10s window from an ECG signal.

2) Normalizing the amplitude values to the range of between zero and one.

3) Finding the set of all local maximums based on zerocrossings of the first derivative.

4) Finding the set of ECG R-peak candidates by applying a threshold of 0.9 on the normalized value of the local maximums.

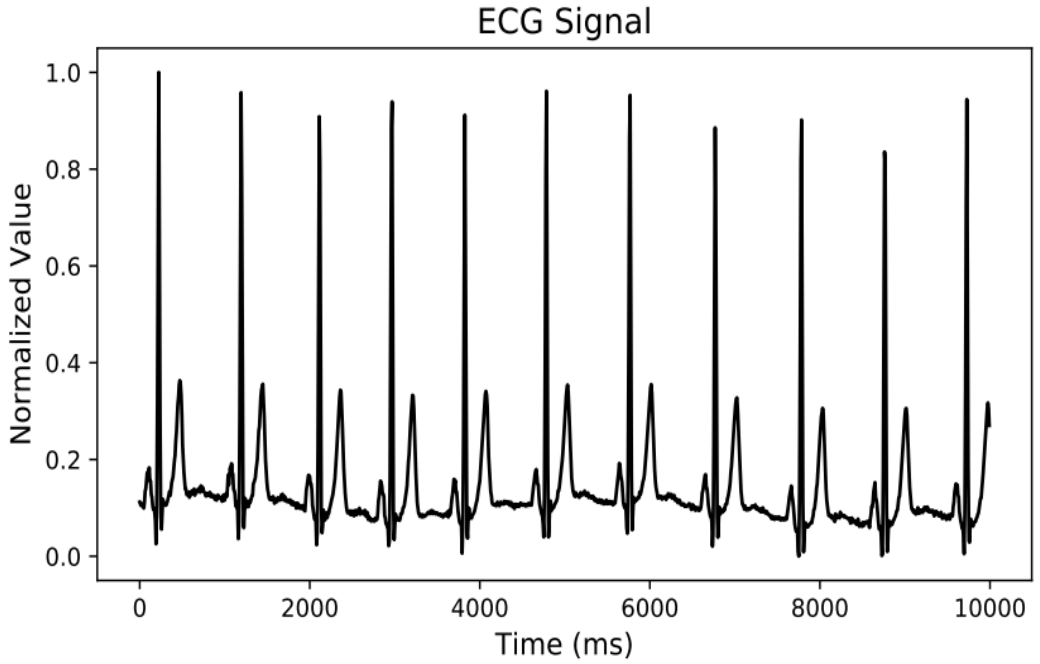
5) Finding the median of R-R time intervals as the nominal heartbeat period of that window (T).

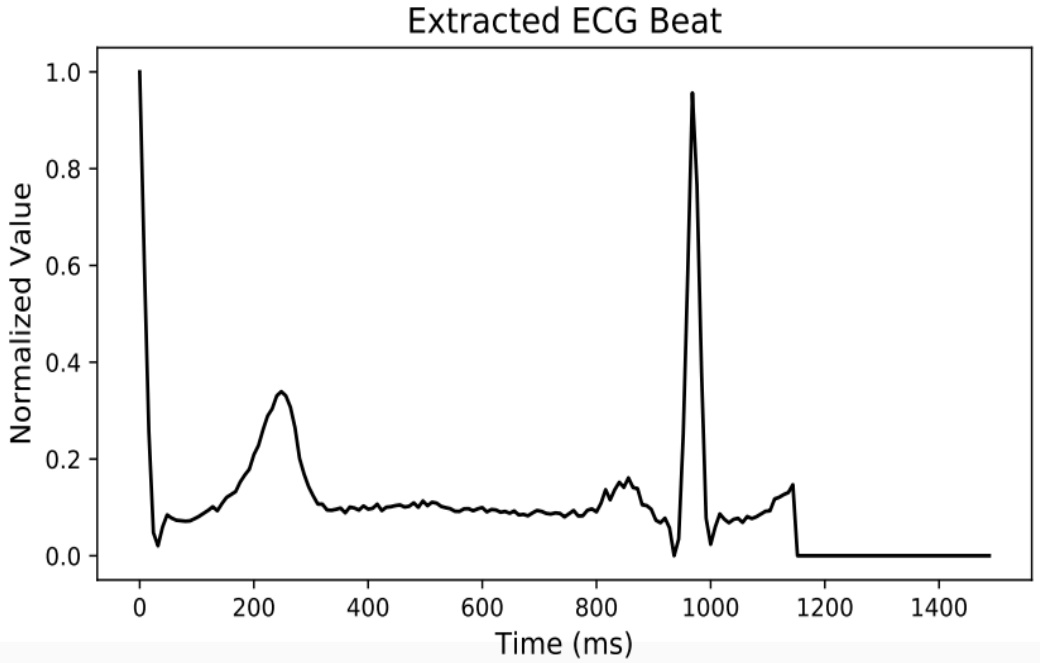
6) For each R-peak, selecting a signal part with the length equal to 1.2T.

7) Padding each selected part with zeros to make its length equal to a predefined fixed length.

***ECG Heartbeat Classification: A Deep Transferable Representation, by Mohammad Kachuee, Shayan Fazeli, Majid Sarrafzadeh**

An example of a 10s ECG window and an extracted beat from it.





My Signal Preprocessing Modifications

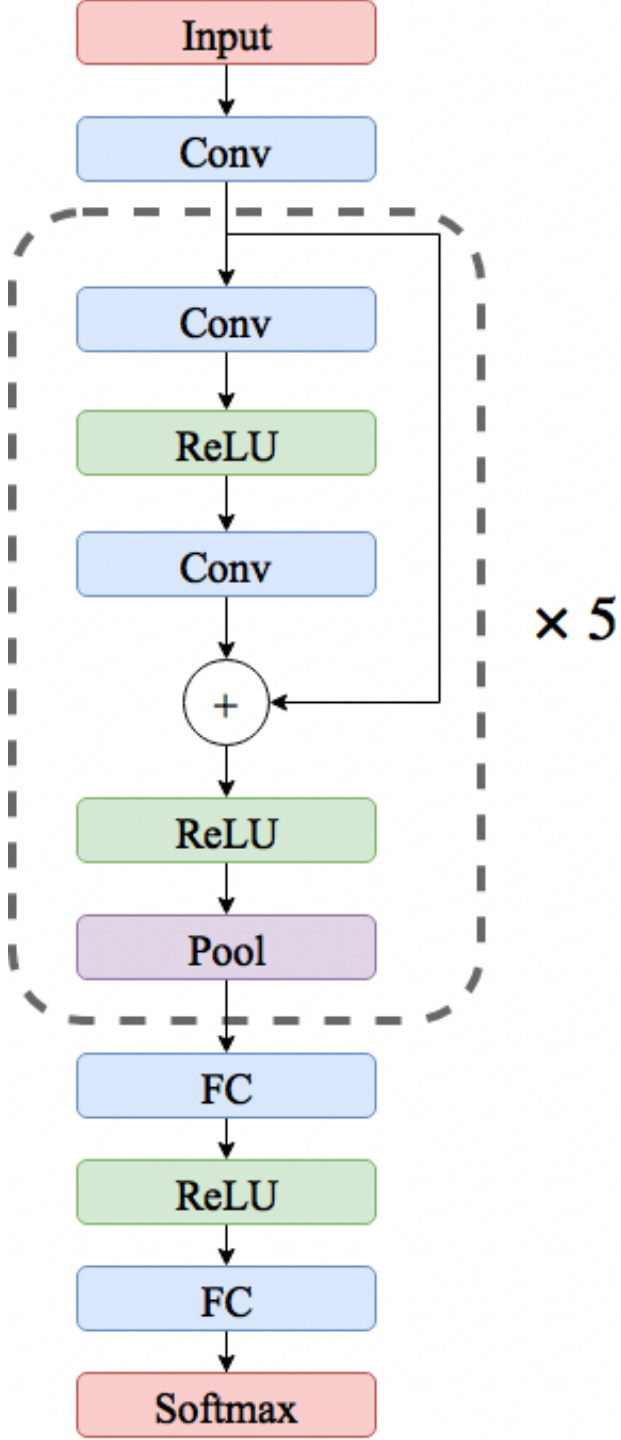
- T = 100 or more, depending on when a Peak arrives after 100x8=800ms
  - Hyperparameter
- Chose Beat Fixed Length = 188
  - This is the length used by the training dataset*

*[5] **ECG Heartbeat Categorization Dataset**
https://www.kaggle.com/shayanfazeli/heartbeat

Machine Learning Model

I followed and implemented the ECG Heartbeat Classification paper's model as it tackles the same ECG problem and classifies the same 5 classes that we are trying to predict.

I trained the model normally using Tensorflow Keras then migrated it to our MCU using Tensorflow Lite.



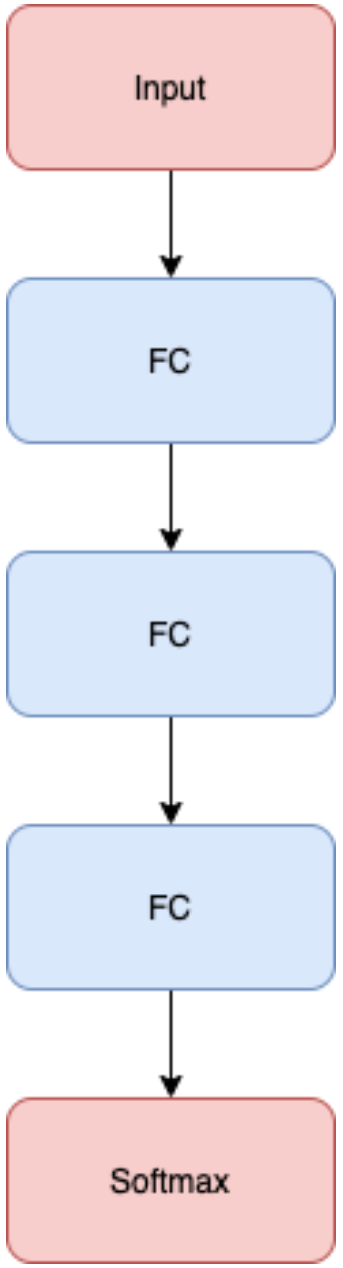
Machine Learning Model

Problem:

RAM size Limitation

Solution:

Changed the Model to 3 Fully Connected Layers



Tensorflow Lite

- After training the model using Keras, it is converted to a TF Lite model using the TensorFlow Lite Converter that generates a FlatBuffer file (.tflite).
- The flat buffer is then optimized using a method called Dynamic Range Quantization which reduces the size of the buffer. The buffer can go up to 4x smaller and can have 2-3x speedup in performance while having a very close accuracy to the original model. This is done by reducing the weights to 8-bit precision.
- Then I included this FlatBuffer in the MCU C project.

```
# Dynamic Range Quantization
converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_quant_model = converter.convert()

# Save the model to disk
open("ecg_model_quantized.tflite", "wb").write(tflite_quant_model)
```

Analyzing the paper model

Complexity: 3537707 MACC
Flash occupation: 171.28 KBytes
RAM: 70.27 KBytes

id	layer (type)	macc		rom	
1	convld_1 (Conv2D)	|	0.8%	|	0.4%
2	convld_2 (Conv2D)	||||||||||||||||||||||||||||||||||	26.7%	||||||||||||||||||	10.1%
4	convld_3 (Conv2D)	||||||||||||||||||||||||||||||||||	26.5%	|||||||||||||||||||	10.1%
5	add_1 (Eltwise)	|	0.2%	|	0.0%
6	activation_4 (Nonlinearity)	|	0.2%	|	0.0%
7	max_poolingld_2 (Pool)	|	0.4%	|	0.0%
8	convld_6 (Conv2D)	|||||||||||||||	13.1%	|||||||||||||||||||	10.1%
10	convld_7 (Conv2D)	|||||||||||||||	13.0%	|||||||||||||||||||	10.1%
11	add_3 (Eltwise)	|	0.1%	|	0.0%
12	activation_6 (Nonlinearity)	|	0.1%	|	0.0%
13	max_poolingld_3 (Pool)	|	0.2%	|	0.0%
14	convld_8 (Conv2D)	||||||||	6.3%	||||||||||||||||||	10.1%
16	convld_9 (Conv2D)	||||||||	6.2%	||||||||||||||||||	10.1%
17	add_4 (Eltwise)	|	0.0%	|	0.0%
18	activation_8 (Nonlinearity)	|	0.0%	|	0.0%
19	max_poolingld_4 (Pool)	|	0.1%	|	0.0%
20	convld_10 (Conv2D)	||||	2.9%	||||||||||||||||||	10.1%
22	convld_11 (Conv2D)	||||	2.9%	||||||||||||||||||	10.1%
23	add_5 (Eltwise)	|	0.0%	|	0.0%
24	activation_10 (Nonlinearity)	|	0.0%	|	0.0%
25	max_poolingld_5 (Pool)	|	0.0%	|	0.0%
27	dense_1 (Dense)	|	0.2%	||||||||||||||||||||||||||||||||||	16.2%
28	activation_11 (Nonlinearity)	|	0.0%	|	0.0%
29	dense_2 (Dense)	|	0.0%	||||	2.1%
30	dense_3 (Dense)	|	0.0%	|	0.3%
31	softmax_1 (Nonlinearity)	|	0.0%	|	0.0%

Analyzing the simple model

Complexity: 12275 MACC
Flash occupation: 6.69 KBytes
RAM: 400.00 Bytes

Compressed x8 because of maximum allowed hex file size

References

[1] **ECG Heartbeat Classification: A Deep Transferable Representation, by Mohammad Kachuee, Shayan Fazeli, Majid Sarrafzadeh**

https://arxiv.org/abs/1805.00794

[2] **TinyML**, by Pete Warden, Daniel Situnayake, ISBN: 9781492052043.

https://learning.oreilly.com/library/view/tinyml/9781492052036/

[3] **Tensorflow Lite documentation**.

https://www.tensorflow.org/lite

[4] **STM32L432KCU Datasheet**

https://www.st.com/resource/en/datasheet/stm32l432kc.pdf

[5] **ECG Heartbeat Categorization Dataset**

https://www.kaggle.com/shayanfazeli/heartbeat