UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA DELL'INFORMAZIONE

"IL CARDIOFREQUENZIMETRO: ANALISI DEL CIRCUITO E IMPLEMENTAZIONE SU ARDUINO"

Relatore: Prof. Matteo Meneghini

Laureanda: Anna Cazzadore

ANNO ACCADEMICO 2022 – 2023

Data di laurea 17/07/2023

Indice:

1. Introduzione

Al giorno d'oggi è possibile misurare i battiti in ogni momento tramite un saturimetro o uno smartwatch, questo è possibile grazie ad una tecnica chiamata fotopletismografia. Questa è una tecnica utilizzata per studiare la diffusione dei raggi luminosi all'interno dei tessuti al fine di studiare il flusso sanguigno all'interno degli stessi. Grazie a questa tecnica è possibile determinare la frequenza cardiaca e la saturazione del sangue in modo non invasivo e in modo molto economico, infatti per assemblare un circuito per effettuare la misura serve solamente una fonte luminosa, un fotodiodo in grado di convertire il segnale luminoso risultate in corrente elettrica e pochi altri componenti per creare dei filtri per amplificare l'ingresso e filtrarlo per eliminare il rumore.

Lo scopo di questa tesi è quello di creare un circuito in grado di misurare sia la frequenza cardiaca che la saturazione del sangue. La prima fase per la progettazione del circuito è la scelta di un'opportuna sorgente luminosa per la creazione del segnale di ingresso e la determinazione delle dimensioni dei componenti necessari per filtrare correttamente il segnale di ingresso e la scrittura di un programma Arduino per poter calcolare la frequenza, l'ossigenazione del sangue e mostrare i risultati della misura su un display. Infine si procede all'assemblaggio del circuito su breadboard per verificare l'effettivo funzionamento del circuito.

2. Componenti del circuito

Lo schematico del cardiofrequenzimetro studiato in questa tesi è mostrato nella Figura 2.1.

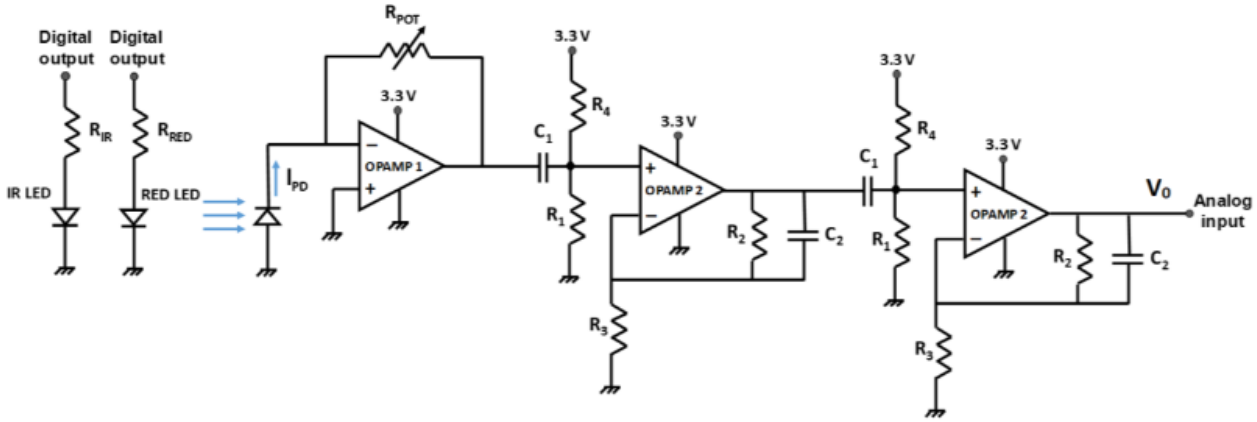


Figura 2.1: Schematico del cardiofrequenzimetro

2.1 Blocco led

La prima sezione del circuito è composta da due led con lo scopo di illuminare il dito e un fotodiodo che ha come obiettivo la conversione della luce che, dopo aver passato il tessuto cutaneo viene riflessa sul fotodiodo, in una corrente. I led sono particolari tipi di diodi a giunzione p-n che sono in grado di emettere luce quando attraversati da corrente elettrica. I fotodiodi sono sempre particolari giunzioni p-n asimmetriche che sfruttando l'effetto fotovoltaico riescono, tramite l'assorbimento dei fotoni, a creare una corrente al loro interno.

Come led abbiamo scelto un led rosso con una lunghezza d'onda pari a 645nm e tensione di soglia pari a 1.75V [2] e come led infrarosso con λ=850nm e tensione di soglia pari a 2.6V [3]. Il fotodiodo scelto è prodotto da Vishay VBP104S [4] ed è sensibile a onde con λ compreso tra 430nm e 1100nm, ovvero sia allo spettro della luce visibile sia alla luce infrarossa. I due led vengono preceduti ciascuno da una resistenza per poter controllarne l'intensità. Impostiamo una corrente pari a I_{RED}=15mA e I_{IR}=3mA. Otteniamo i valori R_{RED} e R_{IR} tramite le seguenti formule

$$R_{RED}=\frac{3.3V-1.75V}{15mA}=103.3\Omega$$

$$R_{IR}=\frac{3.3V-2.6V}{3mA}=233.3\Omega$$

Prendiamo come valori effettivi di resistenze R_{RED}=100Ω e R_{IR}=220Ω.

2.2 Blocco amplificazione e filtraggio

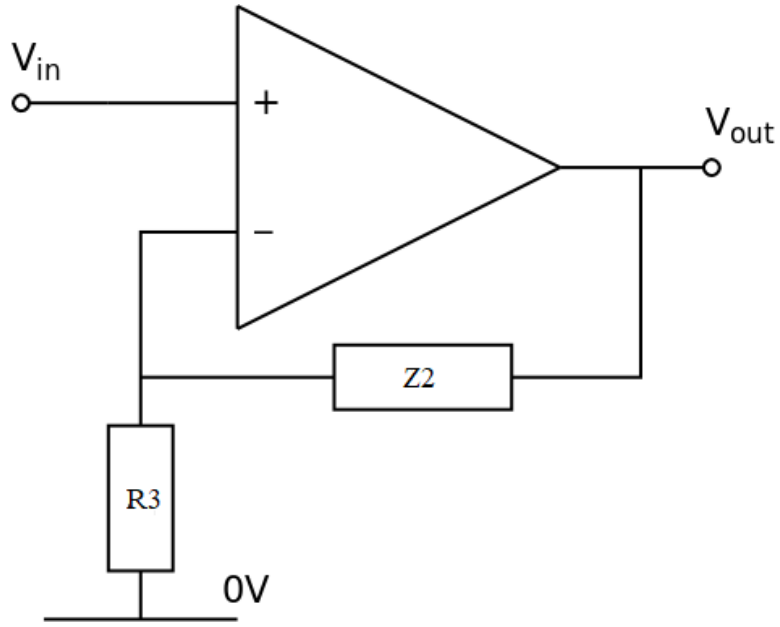
Questa sezione è formata da una serie di 3 filtri e tutti utilizzano un amplificatore operazionale MCP6002. Questo è un amplificatore rail-to-rail, questo significa che l'output può raggiungere le tensioni di alimentazione a meno di un piccolo offset con modulo pari a 25mV per via delle non

idealità dell'amplificatore. Nonostante per lavorare in condizione ottimale l'MCP6002 necessiti di una fonte di alimentazione sia positiva che negativa, noi lo alimenteremo con V_{DD} pari a 3,3V e V_{SS}=0V per non danneggiare la scheda Arduino. Inoltre questo dispositivo è in grado di amplificare un segnale con una banda fino a 1MHz, un intervallo molto più ampio rispetto a quello di interesse per questo progetto.

Il primo stadio è un amplificatore a transimpedenza che ha lo scopo di convertire la corrente generata dal fotodiodo in una tensione e amplificarla. Il filtro è formato da un potenziometro da 100kΩ collegato tra il morsetto v_- e il morsetto v_{out} al fine di aggiustare il guadagno in base al soggetto testato, mentre il morsetto v_+ è collegato a massa.

I due stadi successivi sono identici e hanno lo scopo di filtrare opportunamente il segnale di ingresso al fine di eliminare il rumore e amplificare ulteriormente il segnale nell'intervallo di interesse compreso tra 0,8Hz (48 bpm) e 3Hz (180 bmp). Questo filtro passa-banda è composto dalla serie di un filtro passa-alto in serie ad un filtro passa-basso.

Il filtro passa-basso è composto dal condensatore C2 con capacità di 1µf in parallelo alla resistenza R2 e questi due componenti sono in serie alla resistenza R3 che viene collegata a massa. Questo filtro è ottenuto tramite una configurazione non-invertente in quanto l'ingresso è applicato a v_+ ed è mostrato in figura. Vogliamo che la frequenza di taglio sia a f=3Hz e che il filtro abbia un guadagno in continuo sia pari a 11.



La tensione d'uscita è determinata dalla relazione $V_{out}=(1+\frac{Z2}{R3})$, con Z2=R2//C2=$\frac{R2}{1+sC2R2}$.

La funzione di trasferimento si ottiene dividendo V_{out} e V_-. Da semplici calcoli algebrici otteniamo la formula $W_{LP}=\frac{Vout}{V-}=\frac{1+s*R2\backslash\backslash R3*C2}{1+s*R2*R3}*\frac{R2+R3}{R3}$. Il guadagno in dc è determinato da $\frac{R2+R3}{R3}$ e la

frequenza di taglio è determinata dalla formula $\frac{1}{R2C2}$. Applicando i parametri elencati sopra otteniamo come valori R2=53.05kΩ e R3=5.3kΩ. Questi valori di resistenza precisi non esistono in commercio non esistono per cui prendiamo come resistenze le serie composte da R2=5.6kΩ +47kΩ, R3=4.7kΩ+560Ω.

Il filtro bassa-alto è composto dal condensatore C1 con capacità 10µf e le resistenze R4 collegata all'alimentazione di 3.3V e la resistenza R1 collegata a massa. Il condensatore C1 viene aggiunto per disaccoppiare la componente in continuo, non interessante per questo progetto, da quella di piccolo segnale che deriva dal battito cardiaco. La resistenza R4 viene aggiunta per ottenere un offset in ingresso al morsetto v_+ al fine di evitare la perdita del segnale con componente negativa, in quanto l'intero circuito viene alimentato con tensioni positive. Per questo filtro vogliamo una frequenza di taglio pari a 0.8Hz e che la tensione a riposo in continua al morsetto v_+ sia pari a 0.15V. Nel nostro caso l'uscita che prendiamo in considerazione è quella misurata al morsetto v_+ e l'entrata considerata è v_{in}. L'uscita è determinata dalla formula $v_+=V_{in}\frac{R1//R4}{R1//R4+\frac{1}{sC1}}$. La funzione di trasferimento si ottiene come prima e dopo semplici passaggi aritmetici otteniamo $W_{HP} = \frac{s*R1\backslash\backslash R4*C1}{1+s*R1\backslash\backslash R4*C1}$. La frequenza di taglio viene determinata dalla formula $\frac{1}{C1*R1//R4}$ e il guadagno in dc tramite la formula $v_+=V_{in}*\frac{R1}{R1+R4}$. Applicando i parametri otteniamo R1=20.83kΩ e R4=442kΩ. Prendiamo come resistenze R1=20kΩ e R4 come la serie di due resistenze di valore pari a 220kΩ.

I diagrammi di Bode dei due filtri e del filtro passa banda complessivo sono mostrati nelle figure sottostanti.

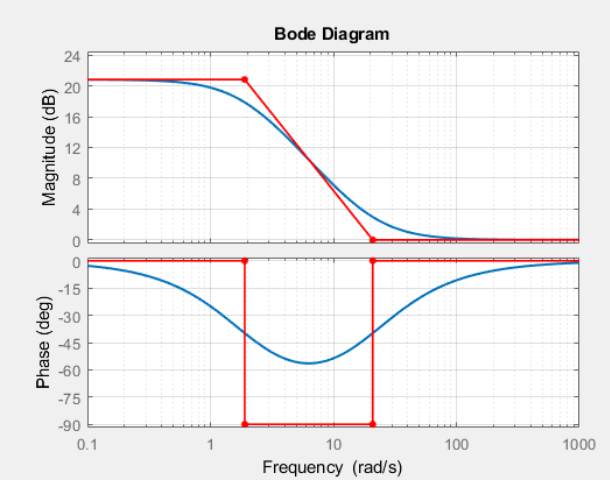


Figura 2.2.1: Il filtro passa-basso

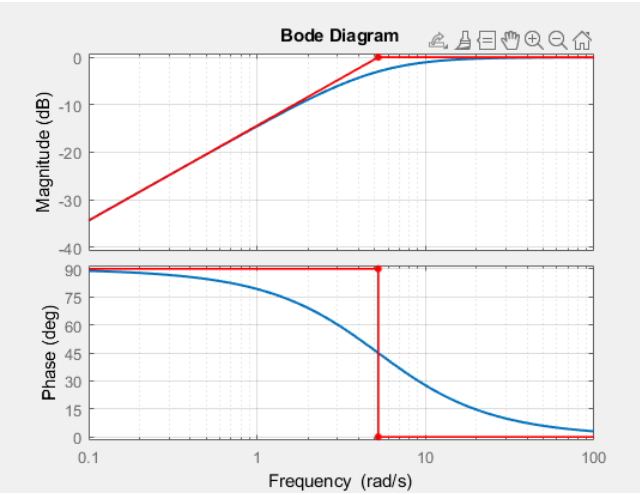


Figura 2.2.2: Il filtro passa-alto

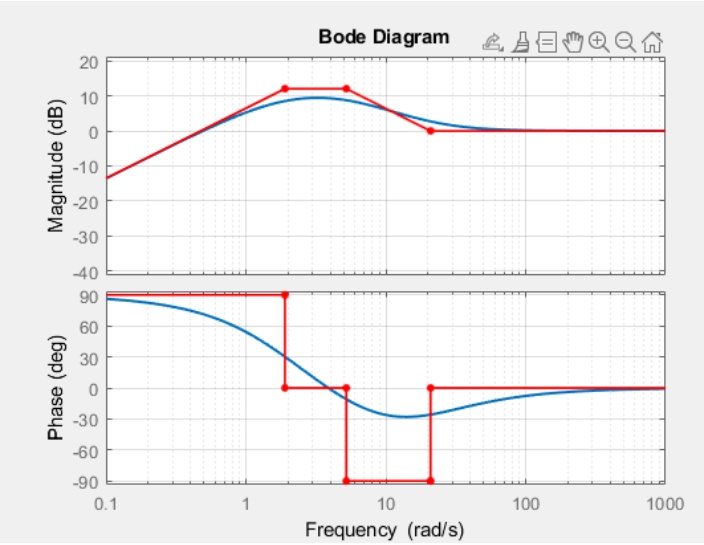


Figura 2.2.3: Il filtro passa-banda

2.3 Blocco visualizzazione

Per visualizzare l'output della misura, la frequenza cardiaca e la saturazione utilizziamo un display LCD TFT Adafruit HX8357. Questo schermo da 3,5 pollici e una risoluzione di 320x480 pixel e può mostrare 262K di colori differenti. Il fatto che questo schermo sia ricoperto da un sottile strato sottile di trasistor, in inglese thin film transistor da cui il nome TFT, riduce i consumi in quanto il campo elettrico viene applicato direttamente nel punto necessario a differenza dei classici schermi LCD in cui la tensione viene fornita dall'esterno del display. Può essere controllato tramite un'interfaccia a 8 bit, ma richiede l'utilizzo di un numero minimo di 12 pin, oppure tramite protocollo SPI, noi sceglieremo quest'ultima opzione. Non è di interesse per questo particolare progetto, ma l'HX8357 è anche uno schermo touch.

2.4 Blocco Arduino

Arduino è una piattaforma hardware a basso costo utilizzata principalmente per la creazione di molteplici tipi di circuiti grazie alla sua versatilità. Per questa tesi viene utilizzata la scheda Arduino Uno. Questa scheda utilizza un microprocessore Artmel ATmega328P a 8 bit e una memoria flash di 32KB, di cui 0.5KB utilizzati dal boot loader. La scheda può essere alimentata tramite porta USB o tramite power jack con una tensione raccomandata compresa tra 7V e 12V. La scheda possiede 14 pin digitali che possono essere utilizzati come pin di input o output tramite codice. I pin 10 (chip select), 11(MOSI) e 13 (SCK) vengono adibiti al protocollo SPI per poter controllare altri dispositivi, nel nostro caso lo utilizziamo per controllare lo schermo. Oltre ai pin sopracitati vengono utilizzati i pin 8 (reset),9 (data/command),12 (MISO) e il pin 5V per l'alimentazione.

Tra questi 14 pin digitali sono presenti 6 sono pin PWM a 8 bit, questo significa che è possibile gestire tramite codice la tensione in uscita dal pin. In questo circuito utilizziamo due di questi pin per accendere e spegnere a comando i due led. Infine sono presenti 6 pin analogici a 10 bit (quindi possono assumere 1024 diversi valori). Nel nostro progetto il pin A0 viene utilizzato come input per convertire la tensione in uscita in un numero compreso tra 0 e 1023 per poter elaborare il segnale in uscita del circuito.

3. Codice Arduino

Questo è il codice Arduino utilizzato per far funzionare il circuito.

Inizialmete importiamo le librerie SPI, Adafruit_GFX e Adafruit_HX8357 che sono necessarie per il funzionamento del monitor. Nominiamo anche i pin di controllo dello schero per una maggiore leggibilità del codice, definiamo delle costanti che rappresentano dei colori in formato RGB e creiamo l'oggetto tft per inizializzare lo schermo HX8357. Definiamo anche i pin di controllo del led rosso (RedLed), del led infrarosso (IRPin) e del pin analogico che utilizziamo per ricevere in input il segnale filtrato e amplificato (heartPin). Oltre a questo definiamo le variabili pixel=1 che utilizziamo come contatore, la variabile maxValue con valore fittizio 0 e minValue con valore temporaneo pari a 10000.

Successivamente definiamo le funzioni necessarie per questo programma.

La prima è la funzione è aquireHeart che ha come input un intero che noi chiamiamo pixel e che legge la tensione al pin heartPin e lo rimappa con un valore compreso tra 0 e 160 al fine di poterlo mostrare nella metà superiore dello schermo e lo salva all'interno dell'array heartValue nella posizione indicata dal valore di pixel.

La seconda è aquireSat, anche questa funzione ha lo stesso input della funzione precedente e come prima legge la tensione al pin heartPin, ma la rimappa con un valore compreso tra 0 e 330, per una maggiore precisione della misura e la salva all'interno dell'array heartValue alla posizione indicata dal valore di pixel.

La terza è printHeart. Questa funzione che come input il numero intero pixel come le funzioni precedenti. printHeart invoca il comando tft.drawLine per tracciare una linea di colore verde a partire della coordinate determinate dal valore predente di pixel e 160 meno il valore salvato nell'array heartValue all'indice di valore pixel-1 e finisce nel punto di coordinate pixel e 160 meno il valore contenuto all'indice pixel in heartValue.

L'ultima funzione si chiama findValues. Questa funzione non ha input e scansiona l'array heartValue per salvare all'interno delle variabili maxValue e minValue rispettivamente il massimo e il minimo valore contenuto nell'array.

Nella sezione di setup definiamo le funzioni dei pin e impostiamo la tensione in uscita di IRPin e RedLed su HIGH per mantenere accesi i due led. Impostiamo lo schermo in orizzontale, coloriamo lo sfondo di nero tramite il comando fillScreen(BLACK) e scriviamo in bianco nella parte bassa del monitor la scritta "Heartrate: ".

Nella sezione loop() iniziamo con un if con condizione pixel<480 che ad ogni iterazione chiama la funzione aquireHeart e printHeart per mostrare in tempo reale il segnale acquisito e incrementa pixel. Questo programma acquisisce un campione ogni 30ms per poter raccogliere campioni per un intervallo di lunghezza sufficiente e nota. Quando la variabile pixel raggiunge il valore di 480

ricopriamo la parte superiore dello schermo con un rettangolo nero, invochiamo la funzione findValue e disegniamo due linee: una blu ad un'altezza di 55 e l'altra di colore giallo ad altezza 75, necessarie per il calcolo dei battiti per minuto. Con un ciclo for riscriviamo i valori dell'arrayHeart value da dei valori compresi tra minValue e maxValue a 0 e 160 per poter visualizzare il segnale in modo più chiaro. Successivamente creiamo la variabile bmp che contiene il numero di battiti. Per evitare errore il valore di bpm viene calcolato in due fasi per ridurre ulteriormente gli errori. Infatti la variabile bpm viene incrementata solamente se il segnale supera la linea blu e successivamente assume un valore inferiore alla posizione della linea gialla. Dopo questi passaggi il valore ottenuto viene moltiplicato per un fattore 4 per poter ottenere l'effettivo numero di pulsazioni al minuto, in quanto la durata della misurazione è pari a 14400ms, circa 15 secondi.
Nell'ultima parte del codice effettuiamo le misure necessarie per il calcolo dell'ossigenazione del sangue tramite la formula S_pO_2=97.94+1.15*R, con R inteso come indice di assorbimento della componente rossa e infrarossa. La formula per calcolare R è $\frac{(\frac{AC\_red}{DC\_red})}{(\frac{AC\_if}{DC\_if})}$, con AC inteso come ampiezza picco-picco di una pulsazione e DC come linea base della pulasazione. Vengono eseguite ulteriori due misure, prima solamente con il led infrarosso accesso e la seconda con il solo led rosso acceso, tramite ciclo for che invoca la funzione acquireSat. Sfortunatamente la memoria della scheda Arduino Uno non è sufficientemente ampia per contenere una matrice 480x3 per cui il codice riscrive ad ogni misura l'array heartValue. Dopo la prima misura della componente infrarossa invochiamo la funzione findValues, con questi valori calcoliamo il denominatore di R. Ripetiamo la stessa procedure, ma con il solo led rosso acceso per calcolare l'assorbimento della componente rossa e di conseguenza il denominaotre di R. Infine creiamo un rettangolo nero nella sezione dove verrà scritto il valore di S_pO_2 per avere la certezza che non ci siano segni dovuti a scritte precedenti e prendiamo li primi 6 caratteri della stringa ottenuta convertendo la variabile SpO2 che contiene il valore della saturazione. Il codice Arduino è riportato qui sotto.

```
#include <SPI.h>
#include "Adafruit_GFX.h"
#include "Adafruit_HX8357.h"

//pin di comando dello schermo
#define TFT_CS 10
#define TFT_DC 9
#define TFT_RST 8

//colori
#define BLACK 0x0000
#define BLUE 0x001F
#define RED 0xF800
```

```
#define GREEN 0x07E0
#define CYAN 0x07FF
#define MAGENTA 0xF81F
#define YELLOW 0xFFE0
#define WHITE 0xFFFF

//inizializzazione schermo
Adafruit_HX8357 tft=Adafruit_HX8357 (TFT_CS, TFT_DC, TFT_RST);

//definizione nome dei pin
#define heartPin A0
#define IRPin 5
#define RedLed 6

int heartValue[480]; //array dove vengono salvati i dati

//definizioni delle funzioni
unsigned long previousTime;
int pixel=1;
float maxValue= 0;
float minValue=10000;

void aquireHeart(int pixel){
  heartValue[pixel]=map(analogRead(heartPin),0,1023,0,160);
}

void acquireSat(int pixel){
  heartValue[pixel]=map(analogRead(heartPin), 0, 1023, 0, 330);
}

void printHeart(int pixel){
  tft.drawLine(pixel-1,160-heartValue[pixel-1], pixel, 160-heartValue[pixel], GREEN);
   }

void findValues(){
  for(int i=20;i<480; i++)
 if(heartValue[i]> maxValue) maxValue=heartValue[i];
  else if(heartValue[i]< minValue) minValue=heartValue[i];
}

void setup() {
  // put your setup code here, to run once:
pinMode(heartPin, INPUT); //input della misura
pinMode(IRPin, OUTPUT);
pinMode(RedLed, OUTPUT);
digitalWrite(RedLed, HIGH);
digitalWrite(IRPin, HIGH);
heartValue[0]=80; //valore medio
tft.begin(HX8357D);
tft.setRotation(1);
tft.fillScreen(BLACK);
```

```
tft.setTextColor(WHITE);
tft.setTextSize(4);
tft.setCursor(10,170);
tft.println("Heartrate:");
tft.setCursor(40,190);

delay(1000);
}
void loop() {
  // put your main code here, to run repeatedly:

if(pixel<480)
{
  if((millis()-previousTime)>30)
  {
    previousTime=millis();
    aquireHeart(pixel);
    printHeart(pixel);
    pixel++;
  }
  }
else if(pixel==480){
  tft.fillRect(0,0,480,160, BLACK);
  findValues();
  tft.drawLine(0, 55, 480,55, BLUE);
  tft.drawLine(0, 75, 480, 75 ,YELLOW);
  for(int i=21; i<480;i++) heartValue[i]=map(heartValue[i],minValue,maxValue,0,160);

  int bpm=0;
  boolean flag=0;
  for(int i=20; i<480; i++) {

  if (heartValue[i]> (160-55)) flag=1;
  if (heartValue[i]<(160-75))
  if (flag){
    bpm++;
    flag=0;
  }
   }
bpm=bpm*4; //t. aquisizione è 15s*4=60s
tft.setCursor(240,170);
tft.print("= " + String(bpm) + " bpm");
 for(int i=11; i<480;i++) printHeart(i);
 tft.setCursor(10,230);
 tft.print("Waiting for SpO2...");
 digitalWrite(RedLed,LOW);
 delay(3000);
int j=0;
 previousTime=millis();
```

```
while (j<480){
  if((millis()-previousTime)>30) {
    previousTime=millis();
    acquireSat(j);
    j++;
  }
 }
 //denominatore componente IR
 findValues();
 float den=(maxValue-minValue)/minValue;

 digitalWrite(RedLed,HIGH);
 digitalWrite(IRPin,LOW);
 findValues();
 delay(3000);
 j=0;
 previousTime=millis();
while(j<480) {
  if((millis()-previousTime)>30) {
    previousTime=millis();
    acquireSat(j);
    j++;
  }
 }
//numeratore componente redled
findValues();
float num=(maxValue-minValue)/minValue;
 float R=num/den;
 float SpO2=R*1.15+97.94;
 tft.fillRect(0, 210, 470, 70, BLACK);
 tft.setCursor(5, 250);
  tft.print("SpO2= " + String(SpO2).substring(0, 5) + "%");
  pixel++;
  }
  }
```

4. Funzionamento del circuito

Le misure sono state seguite nel laboratorio di misure elettroniche del DEI ed è mostrato nella figura 4.1.

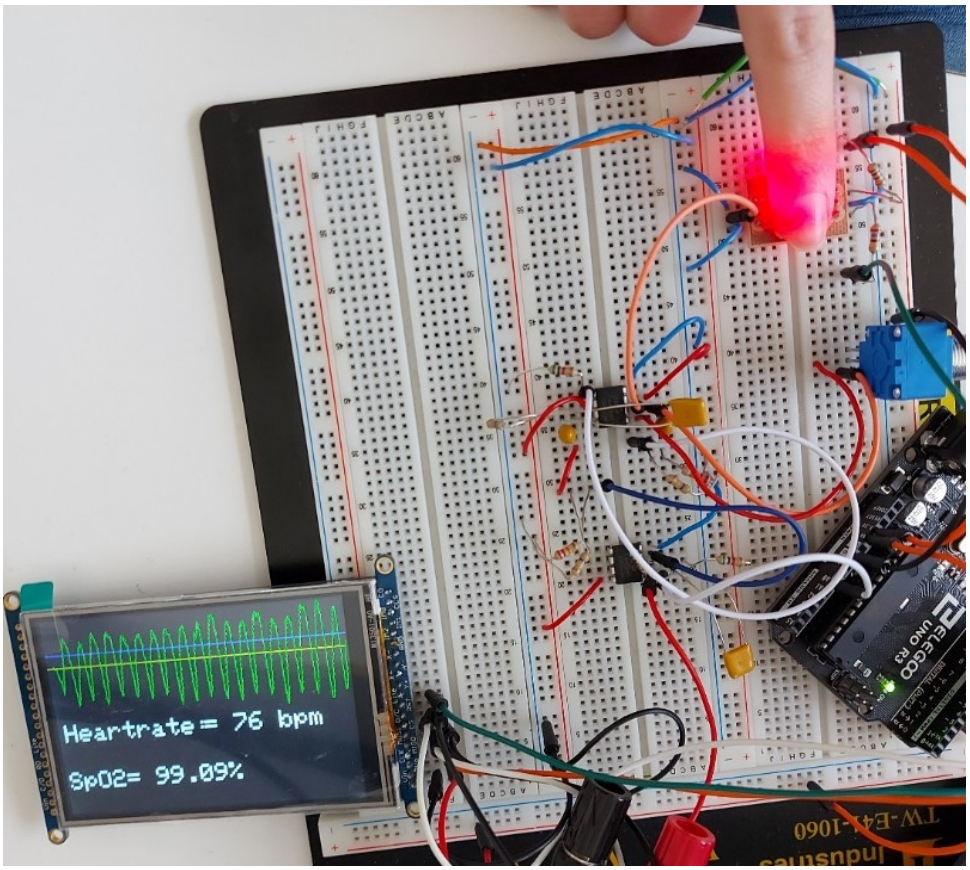


Figura 4.1: Il cardiofrequenzimetro realizzato su breadboard

Inoltre per ottenere una buona misura è necessario applicare poca pressione sul fotodiodo e non muovere il dito per tutta la durata della misura, in quanto questo modifica la quantità di luce che viene ricevuta dal fotodiodo e questo porta ad avere un segnale in uscita con dei picchi che rendono impossibile calcolare correttamente la frequenza cardiaca nonostante il circuito sia funzionante, come è possibile vedere nella figura 8. Inizialmente ho mosso il dito due volte nella prima parte della misura, per poi tenerlo fermo. Come si può notare, dopo un piccolo transitorio, la misura risulta corretta ma per via delle operazioni di mappatura del segnale i battiti non vengono conteggiati correttamente per via della presenza di due picchi. Nei dispositivi commerciali questo problema viene risolto fermando il dito tramite una clip.

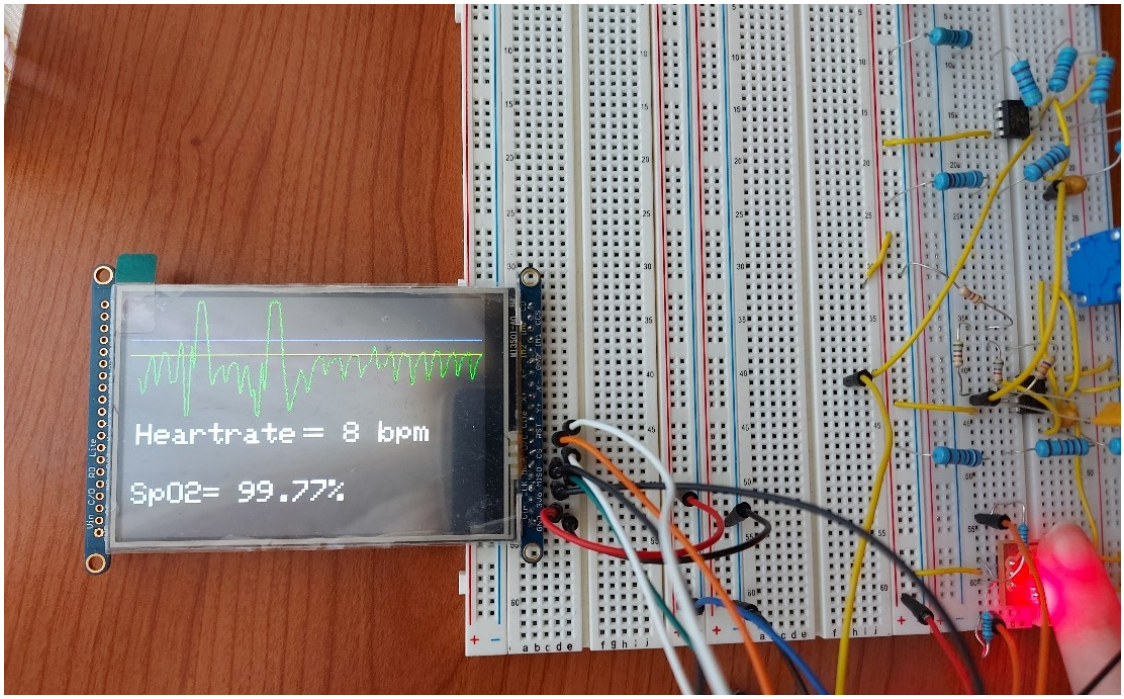


Figura 4.5: Cardiofrequenzimetro realizzato su breadboard, con misura errata

Collegando un oscilloscopio al nodo di uscita possiamo visualizzare meglio la forma d'onda di uscita (figura 9). Ho impostato l'oscilloscopio in accoppiamento ac e confrontando l'immagine mostrata nel display dell'oscilloscopio e quelle sul display Adafruit possiamo notare come la qualità dell'immagine generata in tempo reale è sufficientemente precisa. Inoltre analizzando i dati numerici si può notare che la tensione varia con valori compresi tra -344mV e 281mVe questo conferma che l'offset di 150mV è sufficiente per non perdere pezzi di segnale in quanto quest'offset viene moltiplicato per il guadagno del filtro passa basso, pari a 11, portando l'offset ad un valore pari a 1.65V e quindi i dati risultano coerenti con quello che avevamo progettato e il circuito funziona correttamente senza perdite di informazioni. La tabella contente tutti i dati raccolti dall'oscilloscopio sono mostrati nella tabella 4.1.

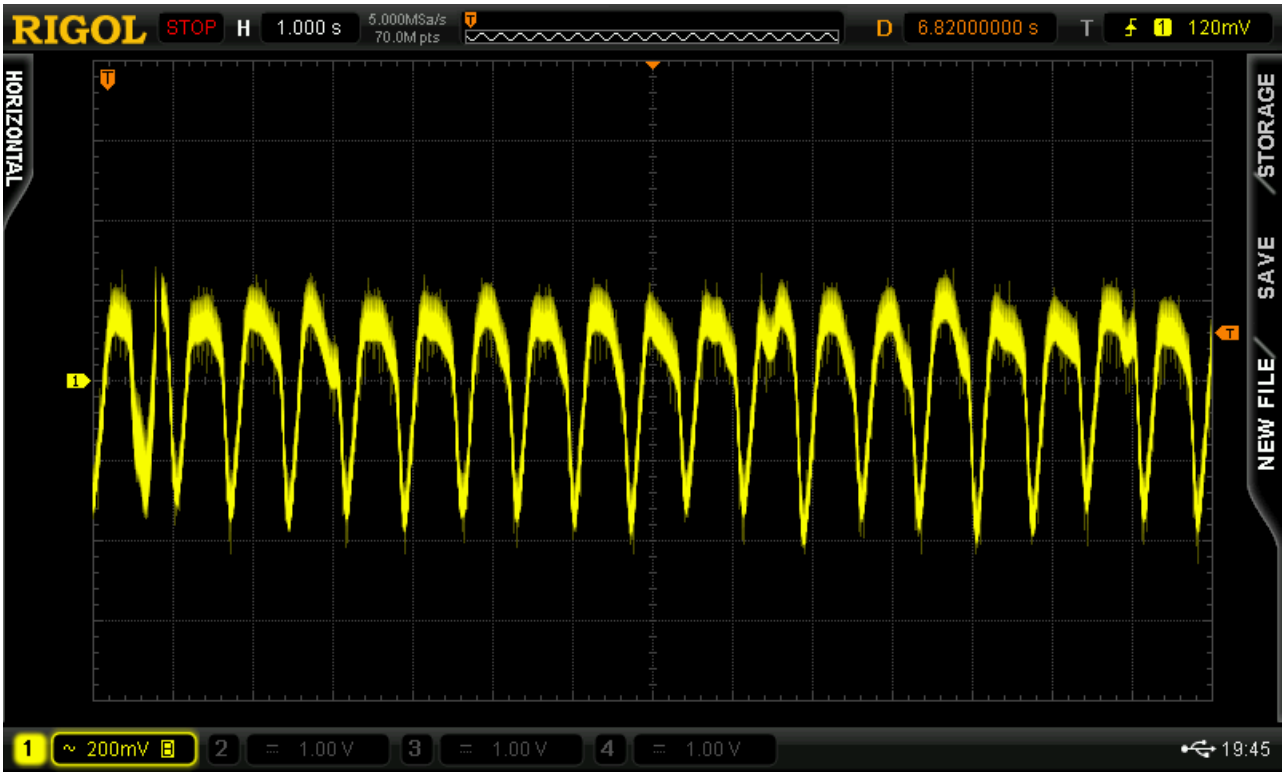


Figura 4.6: Misura tensione al nodo di uscita del cardiofrequenzimetro (componente di piccolo segnale)

Tabella 4.1: Dati oscilloscopio

num. campione	Tensione [V]	num. campione	Tensione [V]	num. campione	Tensione [V]	num. campione	Tensione [V]
0	-0.213	20	0.063	40	0.238	60	-0.069
1	-0.344	21	0.188	41	0.050	61	-0.288
2	-0.188	22	0.050	42	0.206	62	-0.088
3	-0.325	23	0.244	43	0.063	63	-0.325
4	-0.281	24	0.106	44	0.200	64	-0.113
5	-0.150	25	0.225	45	0.069	65	-0.338
6	-0.319	26	0.113	46	0.200	66	-0.344
7	-0.125	27	0.231	47	-0.013	67	-0.125
8	-0.213	28	0.075	48	0.175	68	-0.338
9	-0.019	29	0.238	49	-0.094	69	-0.094
10	-0.156	30	0.113	50	0.075	70	-0.288
11	-0.038	31	0.231	51	-0.169	71	-0.056
12	-0.131	32	0.106	52	0.000	72	-0.250
13	0.031	33	0.231	53	-0.219	73	0.013
14	-0.063	34	0.063	54	-0.250	74	-0.181
15	0.094	35	0.231	55	-0.019	75	0.075
16	-0.013	36	0.100	56	-0.250	76	-0.119
17	0.125	37	0.275	57	-0.031	77	0.119
18	-0.019	38	0.100	58	-0.038	78	0.006
19	0.169	39	0.219	59	-0.263	79	0.281

5. Conclusioni e sviluppi futuri

Il circuito analizzato in questa tesi riesce a misurare i parametri desiderati con sufficiente precisione. Viste che le tensioni e le correnti in gioco risultano molto piccole (qualche centinaio di millivolt al nodo di uscita) il segnale desiderato potrebbe venire coperto dal rumore generato dalle componenti e dalle connessioni con la breadboard, per evitare questo si potrebbe stampare il circuito per ridurre questo tipo di rumore. Oltre a questo abbiamo notato che il circuito è estremamente sensibile al movimento del dito, che può facilmente essere fermato da una clip, ma nei circuiti reali, soprattutto in caso degli smartwatch, non è solamente necessario fermare il dito, ma anche compensare il movimento della mano o del braccio per poter ottenere una misura precisa. L'effetto di questo tipo di errori può essere ridotto aggiungendo un accelerometro al circuito e utilizzare questi dati aggiuntivi per compensare gli errori di misura presenti nel segnale in uscita dal circuito principale.

Inoltre uno studio ha dimostrato come l'utilizzo di un led verde, al posto del led rosso, come fonte luminosa riduce gli errori dovuti al movimento, questo sembrerebbe dovuto al fatto che la luce verde penetra ad una profondità minore rispetto alla luce rossa.

6. Bibliografia

[1] John Allen, *Photoplethysmography and its application in clinical physiological measurement*, 20 febbraio 2007

[2] Osram, *LH T674*, 22 luglio 2017, https://look.ams-osram.com/m/2ec42dcbb6a791a8/original/LH-T674.pdf

[3] Osram, *SFH 4250S High Power Infrared Emitter*, 27 luglio 2021, https://www.osram.com/media/resource/hires/osram-dam-5340617/SFH+4250S_EN.pdf

[4] Vishay, *VBP104S Silicon PIN Photodiode*, 24 agosto 2011, https://www.vishay.com/docs/81170/vbp104sr.pdf

[5] Microchip, *MCP6001/1R/1U/2/4 1 MHz, Low-Power Op Amp*, https://ww1.microchip.com/downloads/aemDocuments/documents/MSLD/ProductDocuments/DataSheets/MCP6001-1R-1U-2-4-1-MHz-Low-Power-Op-Amp-DS20001733L.pdf

[6] *Pinouts*, https://learn.adafruit.com/adafruit-3-5-color-320x480-tft-touchscreen-breakout/pinouts

[7] Jihyoung Lee, Kenta Matsumura, Ken-ichi Yamakoshi, Peter Rolfe, Shinobu Tanaka, Takehiro Yamakoshi, *Comparison Between Red, Green and Blue Light Reflection Photoplethysmography for Heart Rate Monitoring During Motion*, 2013

[8] David Pollreisz, Nima TaheriNejad, *Detection and Removal of Motion Artifacts in PPG Signals*, 08 agosto 2019