BPM FM Heart Rate Monitor MK1

Electronic Design Project 2 ENG2025

Team 16 - BPM FM

James Walker 2793116w, Oscar Petillot 2762520p, Yonghao Wang 2798337w, Andy Mannikum 2811078m

The purpose of this project was to design and build a heart rate monitor from scratch. Working together as an interdisciplinary team allowed us to each contribute towards creating the best product possible. Our main motivations for this project were the satisfaction of designing, debugging and, finally, building a fully working circuit. The competitive aspect of this project also pushed us forward, as we wanted our circuit to be more efficient and user-friendly than our competition.

This project helped us understand how an operational amplifier works in more detail, how to solder on a circuit board and the process of coding in C++ to show the heartbeat trace and frequency in BPM. There were some difficulties across the project regarding the PCB connections on the software and while trying to make the code work. One of the main lessons we learned was to always double check the component values and connections since this did cause some hurdles further down the road.

Acknowledgments

We would like to thank the Course Coordinator, Vincenzo Pusino and the laboratory technician, Andrew Phillips for their guidance and expertise throughout this journey. They have been helpful for our development.

Additionally, we would like to appreciate the help provided by demonstrators Martins Kalnins, Marwan Elfleet, Finlay Smith, for their support during the lab sessions and help to have a better understanding of the concepts.

Everyone's support has greatly improved our academic experience and we are grateful for their contributions to our project.

Table of contents

1. Introduction

We set our design goals to prioritise user experience above all. We believe that this would give the most rewarding benefit than putting any other design goals first as user experience is how well your design ideologies are conveyed to the real world. This defines whether a product is successful or not.

To maximise user experience, we deemed that capability and usability of our board to be the most important factors to achieve this.

With these two factors in mind, the size and power usage of this board take a back seat and will be considered afterwards. This gave us the following initial design focussed on giving the most towards the user:

- 2 8x8 Matrix Displays. (Showing Heartbeat Trace)
- LCD display (Multifunctional Purpose)
- Brightness LED (Heartbeat Pulse via PWM)
- 7 Segment Display (Showing BPM)
- 'No Pulse' Detection
- Slide Switch (To Change Modes on the Board)

A stacked PCB was also considered with the top layer containing all the peripherals but was dismissed to keep with deadlines.

2. Methodology

2.1. Principle of operation

The sensor is composed of an LED and a phototransistor. When the finger is placed in the sensor, the finger will block the light emitted by the LED. The more light is blocked, the lower the voltage. The less light is blocked, the higher the voltage, because when the human body produces a pulse, the blood flows back, so less light is blocked, that is, when the voltage peak is generated.

2.2. Overall diagram

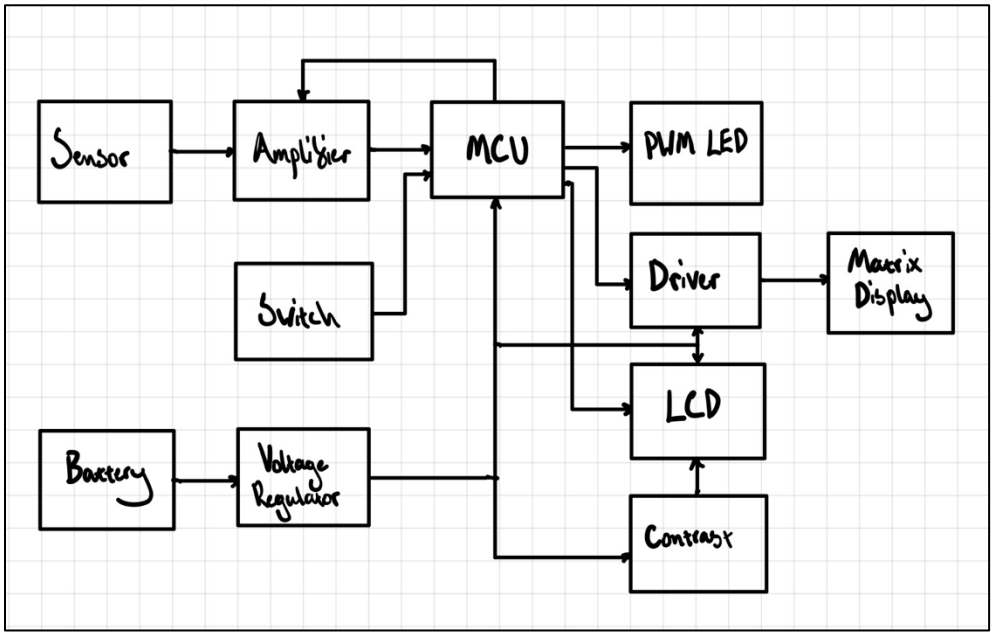


Figure 1. represents the heart rate monitor in terms of a block diagram.

2.3. Sensor

The sensor is composed of an LED and a phototransistor. When the finger is placed in the sensor, the finger will block the light emitted by the LED. The more light is blocked, the higher the voltage. The less light is blocked, the smaller the voltage across the phototransistor. As blood flows through the finger, less light is transmitted through it thus making pulses detectable. Peaks in blood flow can be measured as that is when the least amount of light is transmitted, creating a voltage peak. The phototransistor was originally designed as a detector in industrial for electronic circuitry, measurement, and control. It is being applied to detect changes in light as blood flows.

2.4. Amplifiers

The capacitor inside the sensor is used for filtering. The sensor outputs a millivolt model and enters the operational amplifier. It first passes through a high pass filter and then two low pass filters, and finally outputs an amplified signal that can be detected.

The circuitry integrates low-pass and high-pass filters with a focus bandwidth of 1-3Hz, which will improve the performance of the device by eliminating unwanted noise, especially mains noise at 50Hz. In addition, the amplification of the signal is critical to bring the subtle output of the sensor, which is in millivolts, up to a scale that the ADC can read. Therefore, a gain of 237 was designed for consistent and accurate filter circuit output.

2.5. Microprocessor

We used the Mbed LPC1768 board due to its wide range of features, such as SPI, Analogue input and output signal, integrated power regulation and PWM output. The Mbed is also inexpensive and widely available. Its ability to support various peripherals enabled us to seamlessly integrate and experiment with the sensor and communication modules. This led to rapid prototyping and development processes. In addition, the Mbed offers ease of software integration because of its USB drag 'n' drop feature.

However, the Mbed is an embedded processor, meaning it can only complete one task at a time. There are also a limited number of pins for connecting to displays, so it is important to minimise the number of direct connections to the board using components such as drivers.

2.6. Display

LCDs were chosen for their low cost, low power consumption, usability and wide range of applications. LED matrices are favoured for their small size, long service life, compatibility with integrated circuits, and low power consumption. Display information, not only traces but also images, the display chip makes the project more challenging. Choosing a universal cathode display driver facilitates communication and drastically reduces the number of pins connected to microcontroller. The microcontroller unit has three connections with the matrix display; a clock,

MOSI, and load signal. Overall, display driver chips contribute to simplifying projects by minimizing complexity and connectivity requirements.

2.7. PSU

Our design goal was the user experience. It was not our priority to minimise power consumption at the beginning, as we decided that we wanted to design our power supply unit around the display components, and not the other way around. More parts means that the power supply to be used may be larger.

Design calculations were performed to the battery selection, considering the expected current consumption of components such as the LED matrix, Mbed, sensor, PPG LEDs, analogue amplifiers, and LCD screens.

LED matrix display calculation:

1 row of 8 LEDs = 80 mA, 1 LED = 10mA, 1 row of 8 LEDs = 80 mA

1 row of LEDs at a time, so total current absorption = 80 mA

Mbed microcontroller operating current = 10mA

Pulse sensor PG LED operating current = 10mA

Analogue op amp operating current = 1 mA

LCD screen operating current = 1 mA

MCU operating current = 140mA

Total current requirement = 10mA + 80mA + 10mA + 1 mA + 1 mA + 140mA = 242mA

Calculated resistance = V/I =5V/242ma = (approx.) 21 Ohms

Based on these calculations, a resistance of 21 ohms was determined. Referring to the 100-Ohm resistance curve, the battery life should be around 8 hours for a 100-Ohm resistance, however we predict the battery life to be around 2.5 hours due to our 21-Ohm resistance.

3. Software Design

3.1. Pulse Rate Measurement Algorithm

To determine the pulse rate, we chose to compare the signal to the rolling average. As the signal oscillates from above and below the rolling average, we can define a single wave thus making pulse rate available to calculate.

The pulse measurement code '**bpmRead**' shown in Fig. 2 reads the signal and compares it against the rolling average, incrementing a counter each time. Single waves are defined every

rising edge (when the signal goes from below the rolling average to above it), when this is done the counter gives the period multiplied by a constant analogous to the time for one loop of the main code to be executed. The counter value is then appended onto an array containing the previous 10 wave counter values where the average of these values is calculated. Once this is done, the actual pulse rate is determined by multiplying this number by a constant. The average is done in the code '**bpmAvg**', it executes the following:

$$Actual\ BPM = constant * avg(BPMCounter)$$

This constant was calculated by dividing the counter value with the actual BPM given by a smart watch, we found this number to be roughly 275. The constant had to be calculated after the final iteration of code was completed since the counter increments according to the speed of the main code.

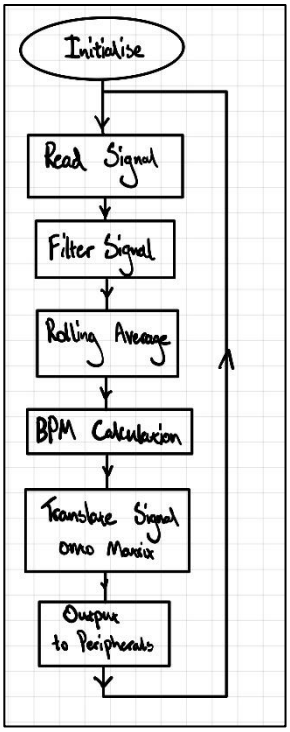
```
135  int bpmAvg(){//Average BPM over 10 samples
136      for (int j = 0; j < 9; j++){ //shift every element to accomodate for next value
137          bpm[j] = bpm[j+1];
138      }
139      bpmInt = 0;
140      for (int j = 0; j < 9; j++){
141          bpmInt = bpmInt + bpm[j]; //bpmInt - intermediate float value.
142      }
143      bpmInt = bpmInt/10; //divide by 10 to complete average sum
144      bpmInt = bpmInt * 275; //multiply value with constant to give BPM
145      bpmOut = bpmInt;//Convert to integer - LCD printf float to 0dp keeps "f" after number
146      return bpmOut;
147  }
148
149  void bpmRead(){//Read trace to sample BPM
150      if(gpo == 0){//if currently low
151          if(i > rAvgOut){//transition to high = rising edge
152              gpo = 1;//set to high
153              bpm[9] = 100.0/bpmCounter; //full cycle - set bpm
154              bpmCounter = 0;//reset counter for next wave
155              bpmAvg(); //gives average pulse rate of last 10 waves - it returns 'bpmOut'
156          }
157          else{//still low
158              bpmCounter++;
159          }
160      }
161      else{//else currently high
162          if(i > rAvgOut-0.1 ){//still high - '0.1' for histeresis
163              bpmCounter++;
164          }
165          else{//transition to low
166              gpo = 0;
167              bpmCounter++;
168          }
169      }
170  }
```

Figure 2: Code for functions 'bpmRead' & 'bpmAvg' which output the variable 'bpmOut' containing the average pulse rate of the last 10 heart beats.

3.2. General Structure

The general structure of the code comprises of a main branch shown in fig. 3 which calls helper functions according to the flowchart in fig. 4.

```
272  int main(){
273      setup_dot_matrix ();      /* setup matric */
274      LedPWM.pulsewidth(0.005);  /* set PWM */
275      LedPWM.period(0.001);
276
277      while(true){
278
279          rawAin = Ain; // Take sample
280          i = (alpha*rawAin)+(1-alpha)*rollingAverage[199]; //filter sample
281
282          rAvgOut = rAvg(i);//find rolling average
283          i = i - rAvgOut; //take out average from signal before outputting
284          bpmRead();//find pulse rate - outputted to 'bpmOut'
285
286          i = (int(i * 8))/8.0;//place signal on a level 0.125 between 0-1
287
288          Aout = i; //output signal to DAC - redundant on PCB
289          LedPWM = i; //set LED brightness
290
291          wait_us(1900);//slow code down (imitates a printf command)
292          matCounter++;
293          if (matCounter>15){ //roughly 16Hz = (1/avgBPM * columns on 8x8 Display)
294              matrixLevel();//fill heartbeat_output - 8x8 data with gaps
295              wogaps();//fill heartbeat_wogaps - 8x8 data w/o gaps
296              output();//output data to peripherals
297              matCounter=0;
298          }
299      }
300  }
```



3.2.1 Read & Filter

After initialisation, variable '**i**' is set to the value of the waveform through an '**ADC**' and is put through a 1st order filter designed to get rid of any unwanted noise spikes.

3.2.2. Rolling Average

DC drift is removed through the '**rAvg**' function. '**rAvg**' takes the filtered '**i**' value and its previous 200 values from past loops, the average of these values is essentially the DC offset caused by DC drift. On initialisation, there are no previous values, so the function waits until 200 values have been collected before calculating the average.

3.2.3. BPM Calculation

The BPM/ pulse rate is calculated in the '**bpmRead**' & '**bpmAvg**' functions and gives the average BPM of the previous 10 heartbeats in the variable '**bpmOut**'.

3.2.4. Rounding Signal

The '**DAC**' takes a value between 0-1 so variable '**i**' is rounded to one of 8 levels on a scale of 0-1. The DAC is on the final PCB is not used however the line of code '$Aout = i$' remains for

debugging purposes. We need to output to the 8x8 LED matrix, so it is useful to round '**i**' to 8 levels.

The first peripheral, the PWM LED, is set. Since '**i**' in the same format used for setting the duty cycle, the LED outputs a certain brightness corresponding to its value.

3.2.5. Translate Signal

To send data to the 8x8 LED matrix, we use the 'pattern_to_display' function with an 8-element array . These 8 elements each represent an 8-bit number which corresponds to the lighting LEDs on the columns and rows on the matrix. Knowing this, we find the binary number using the following formula: $Binary\ Number = 2^{(i*8)}$. The binary number (value 2^n) is then appended to the 8-element array which when sent to the matrix. In this form, sharp increases in the signal such as the main pulse of a heartbeat, large gaps form which need to be filled in. Filling in the gaps in the matrix requires executing the following formula:

$$\text{Gaps to fill (Binary Number)} = \sum_{k=(i*8)}^{[i-1]*8} 2^k$$

Where 'i' is the current signal and 'i-1' is the previous signal.

The number calculated is then added onto the same element in the matrix array. Fig. 5 gives a diagrammatic explanation, where 'level' and 'power' are analogous to each other. Outputting this array gives a clean trace.

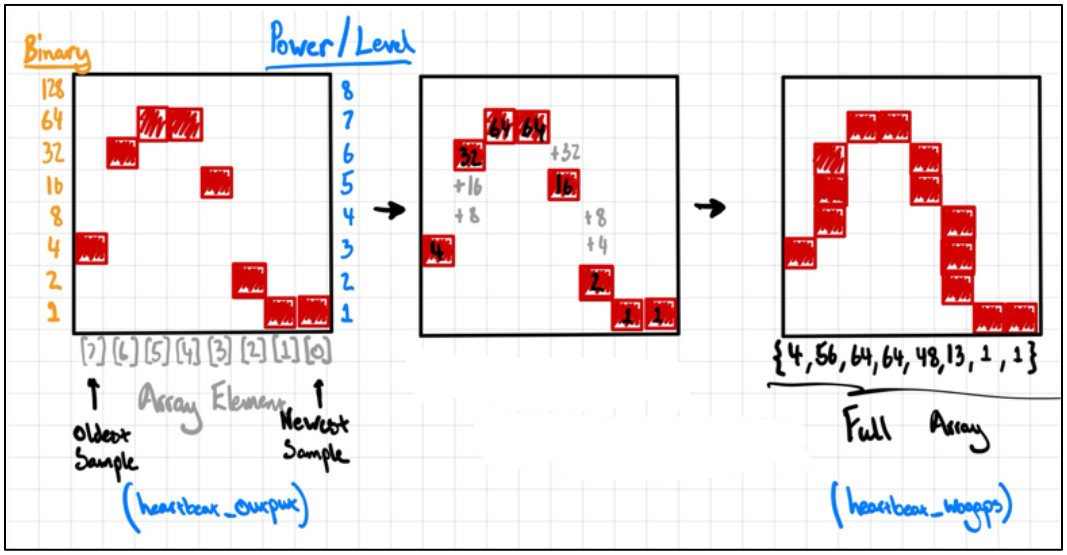


Figure 5:

(a): the array 'heartbeat_output' if outputted to the matrix. It contains the levels of the signal however gaps need filled.

(b): A diagram of the intermediate steps to calculate the binary values of the LEDs in the gaps.

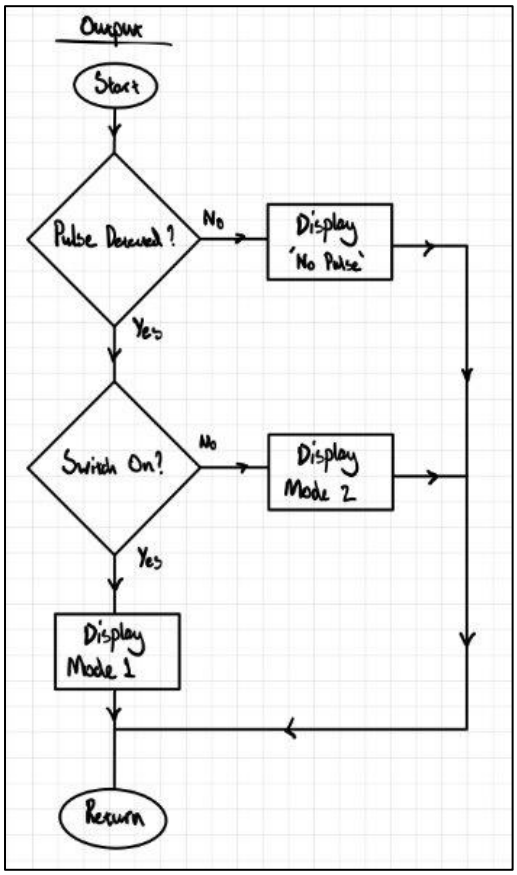
(c): the binary values of the gaps added with the levels creates a complete array ‘heartbeat_wogaps’ which is outputted to the matrix.

3.2.6. Output to Peripherals

When outputting to the peripherals, the function ‘output()’ is run which follows the flowchart in fig. 6.

‘output()’ begins by running function flatline(). This checks if all the elements in ‘heatbeat_output’ are equal – this indicates no pulse is being detected and therefore should be outputted to the displays instead.

If a pulse is detected, then depending on the position of the slide switch, one of two configurations of display modes are outputted to the peripherals.



4. PCB/Breadboard and actual device

To optimise the efficiency of our circuit board, we wanted to minimise the number of vias, as they can introduce problems such as parasitic capacitance, inductance and resistance that can signal quality and increase noise[1]. Having many vias would have also increased the manufacturing time as there would be much more soldering to be done.

Our first version was not very well optimised, as all the components were placed straight away, making it very hard to navigate and know how to connect everything up in an efficient manner. We decided it was best to start from scratch, placing only certain components at a time. This allowed us to come up with a much cleaner way of organising our PCB.

To minimise the number of vias, we made sure to place components that were close by on the schematic as near as possible to each other on the PCB. This also avoided very long connections that cross the entire board, which therefore minimises the resistance that the signal experiences. We also took advantage of the ability to route connections to and from resistors and capacitors on the top, which also reduced the amount of vias, and the distance between connections. In our final circuit, we only had 26 vias, compared to well over 70 in our first version.

The option to mount certain components on the bottom of our PCB also helped. We decided to bottom-mount the battery holder on out board to save space, as well as keep the top of the board aesthetically clean, with the top mainly covered by displays.

Due to the configuration of the op-amp circuit, noise pick-up was not a problem, so it was not a priority in terms of orientation of the components. For example, having parallel traces in the analogue part of the circuit can introduce crosstalk, which is a form of noise caused by electromagnetic fields[2]. We minimised this by avoiding traces running parallel to each other in this part of the circuit.

---

[1] https://www.linkedin.com/advice/1/what-advantages-disadvantages-using-vias-planes-ground#:~:text=Vias%20are%20essential%20for%20routing,quality%20and%20increase%20the%20noise.

[2] https://www.techtarget.com/searchnetworking/definition/crosstalk#:~:text=Crosstalk%20is%20a%20disturbance%20caused,has%20a%20varying%20electromagnetic%20field.

5. Results

Like almost every engineering project, we encountered drawbacks along the way. One of the first problems we encountered was a difficulty in reading smaller heartbeat signals. We solved this by changing some of the resistor values in our operational amplifier circuit in order to increase the gain, therefore increasing the small signals to an amplitude at which they could be read. Due to the high gain, and the difference in heartbeat from person to person, there were cases where the second, smaller pulse was read as a maximum pulse, which would render our BPM readings inaccurate. The easiest way to combat this was to adjust the heart rate sensor itself so that it was looser for stronger heartbeats. The best way to test this was by having as many different people try our sensor to see how it would react to different people's heartbeat signals.

During the PCB design process, we struggled with fitting every component onto the board due to limited space. We decided to remove one of our two 8x8 matrix displays as this component and its drivers were not necessary but took up a large proportion of the board. This made the board to be much more efficiently connected.

Once the board was manufactured, we discovered that the battery holder pins were the wrong way round and realised that it had been placed wrong on the schematic. To solve this issue, we used two small, insulated wires to invert the connections of the pins. While not ideal, this method avoided flipping the orientation of the battery holder, which would have resulted in it poking out of the bottom of the board.

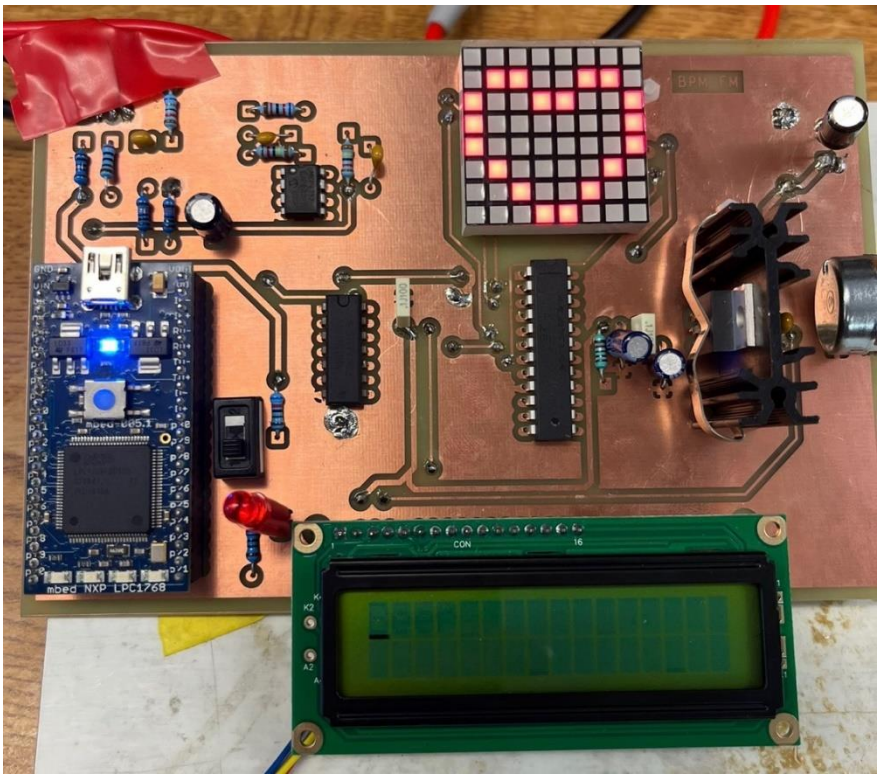
The final design of our board utilised an 8x8 matrix display, LCD, and a variable brightness LED to show the heartbeat signal, as well as a switch the change between the set functions of the displays.

Two modes were implemented, as shown in the table below:

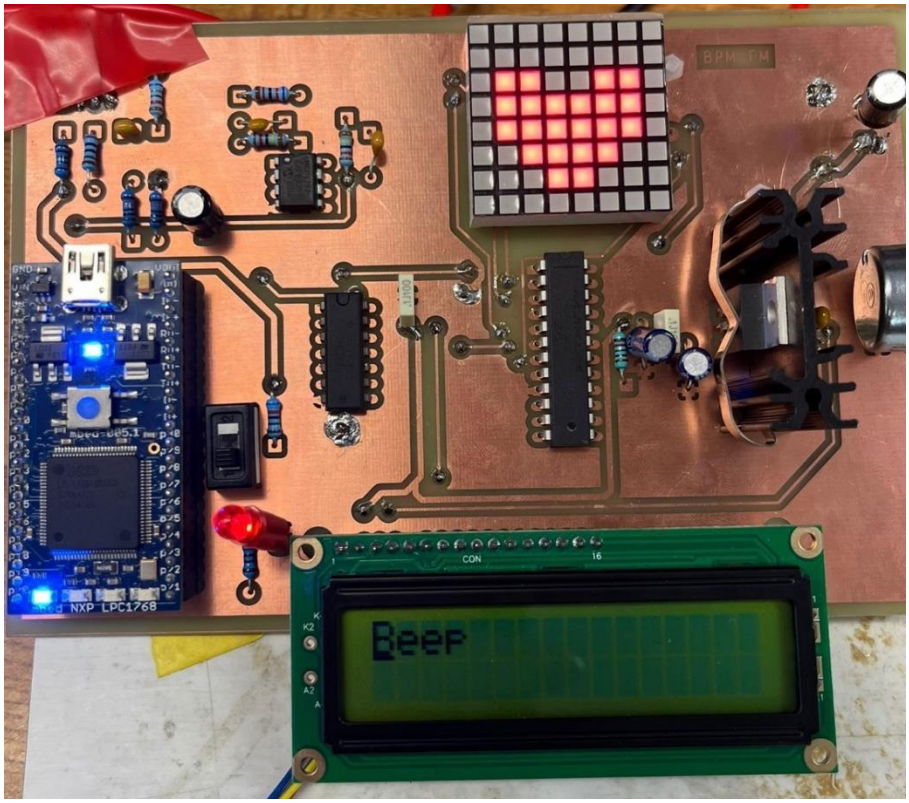
MODE 1 - Pulse/Animation	MODE 2 - Trace/BPM
Brightness LED (Signal Intensity)	**Brightness LED** (Signal Intensity)
LED Matrix (Heart Beating)	**LED Matrix** (Signal Trace)
LCD (display when pulse detected)	**LCD** (display BPM in real-time)
MCU LED (Heart Beating)	

The following pictures show Mode 1 working:

Signal Minimum (LED at minimum brightness, nothing displayed on LCD, open heart animation on matrix display)

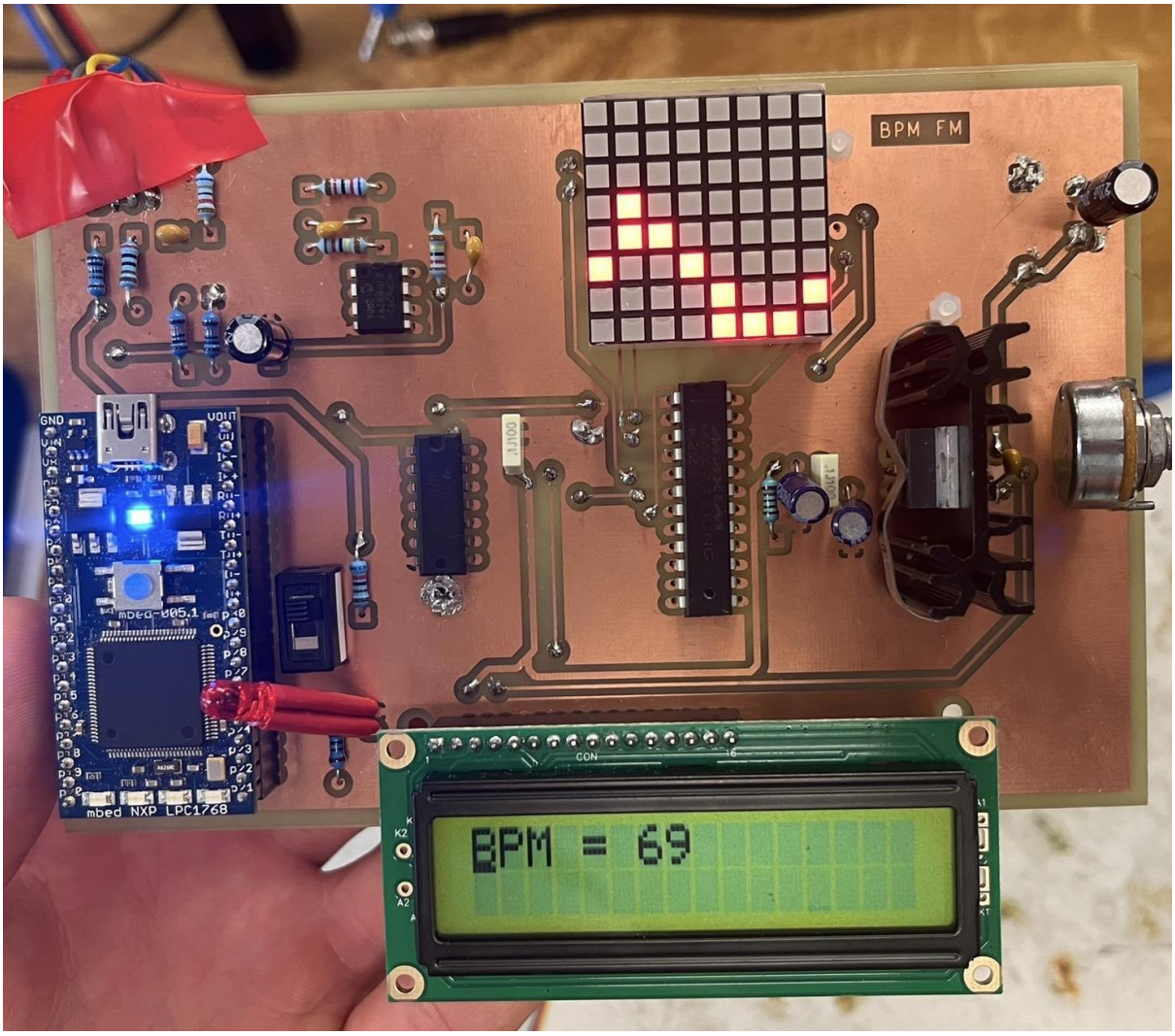


Signal Maximum (LED at maximum brightness. 'Beep' displayed on LCD. Closed heart animation on matrix display. LED 1 on microcontroller on at signal peak.)



The following picture shows Mode 2 working:

Matrix Display showing heartbeat trace. BPM of heartbeat displayed on LCD. LED varying in brightness with the signal.



While we did have to remove some components from our original plan, we made sure to use every component to its maximum capability. This resulted in our circuit having more features than originally planned, as we believed we should not leave components unused between modes.

In terms of software, the matrix display originally followed the trace from left to right, with the oldest sample on the right. However, we changed this to match how it would be on an oscilloscope trace, therefore following the heartbeat trace from right to left, with the oldest sample on the left.

The circuit drew 275mA from the power supply.

6. Conclusion

In conclusion, we managed to design and build a fully working heart rate monitor circuit that utilised all the components to their maximum, creating an enticing user experience which offers all the necessary functions of a heart rate monitor, and so much more. We prioritised usability and capability for our project, but if we had more time, we would have also focussed on how to reduce the power consumption of our circuit. Learning from our competition, the use of a buck regulator would eliminate the need for a heatsink, as much less heat would be dissipated, and it would be negligible.

Features that could be added to the circuit to improve its usability would include a speaker that beeps when a heartbeat is detected which could be used to alert if a patient is flatlining. Also, a type of sensor to detect whether there is a finger in the heartrate sensor could be implemented, to avoid confusion between a patient flatlining or no finger in the sensor.

One thing we learned is to always double check every part of the process along the way, such as connections and component values, as even the smallest of errors can come back and cause significant problems further down the road.

Although our final design is not what we had originally envisioned, we are all satisfied with the result of this project.

7. Glossary of acronyms

PCB – Printed Circuit Board
LCD- Liquid Crystal Display
PWM- Pulse Width Modulated
BPM- Beats per Minute
LED- Light Emitting Diode
SPI- Serial Peripheral Interface
MCU- Microcontroller Unit
DAC- Digital Analogue Converter
ADC- Analogue Digital Converter
Op-Amp- Operational Amplifier
USB – Universal Serial Bus

8. References

LCD Datasheet: https://www.sparkfun.com/datasheets/LCD/ADM1602K-NSW-FBS-3.3v.pdf (Date Accesed:23.03.2024)

Battery Power source: https://www.duracell.com/wp-content/uploads/2020/02/9V-Duracell-Plus.pdf (Date Accesed:23.03.2024)

Red LED L7113: https://www.kingbright.com/attachments/file/psearch/000/00/00/L-7113ID(Ver.29A).pdf (Date Accesed:23.03.2024)

Phototransistor BPV11: https://www.vishay.com/docs/81504/bpv11.pdf (Date Accesed:23.03.2024)

MCU LPC1768: https://os.mbed.com/platforms/mbed-LPC1768/ (Date Accesed:23.03.2024)

8x8 LED Matrix: https://components101.com/displays/8x8-led-matrix-module (Date Accesed:23.03.2024)

Heatsink: https://datasheet.octopart.com/LM2937ES-5.0/NOPB-National-Semiconductor-datasheet-9670594.pdf (Date Accesed:23.03.2024