Nolan Knight
BME595 – Bioinstrumentation
Dr. Ken Yoshida
April 22nd, 2024

Table of Contents

Summary of Device:

This device aims to trigger motor movement every instance of an individual's heartbeat. For further verification, an LED on both the circuit board and microcontroller will flash when a heartbeat is registered by the microcontroller. An image of the assembled device can be found in the Appendix.

Theory of Operation:

Introduction:

This section covers the theory of operation for both the hardware and software of the developed instrument. This is done by diving into the architecture of the physical system and C code implemented in the MSP430 microprocessor. The overall operational design is supposed to record the ECG of a patient and produce a digital output which goes to the high state at every R peak in the patient's heart rate. This digital signal will control a motor connected to a motor driver, with the input pin being connected to the digital output pin on the MSP 430, thus when the digital pin is at its high state, the motor will spin.

Hardware:

System Architecture:

The system architecture for the hardware component of the instrument is broken into three components. The first is an MSP430 microcontroller, the second an ECG module chip, and the third a motor driver chip. The specific model of the MSP430 is the FR2355 and the pin diagram is shown in Figure 1.

Figure 1: MSP430-FR2355 Pin layout

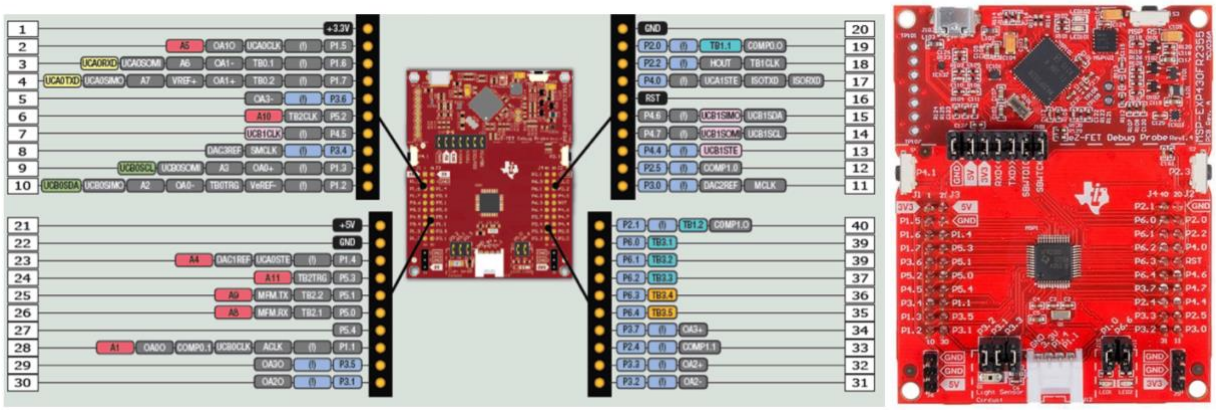


Figure 1: MSP430 microprocessor pin diagram and image of FR2355 model.

The instrument is designed to process an analog signal received by the MSP430, which is an input signal from the ECG module chip. The specific chip used in the instrument is produced by Sparkfun, and the model is the AD8232 Single lead heart rate monitor. This chip will be powered by one of the MSP430's 3.3V power pins. Its output is an analog signal representing the electro potential of the heart during its cycle (ECG). Figure 2 depicts the pin layout of the ECG module chip.

Figure 2: AD8232 Single Lead Heart Rate Monitor Pin layout

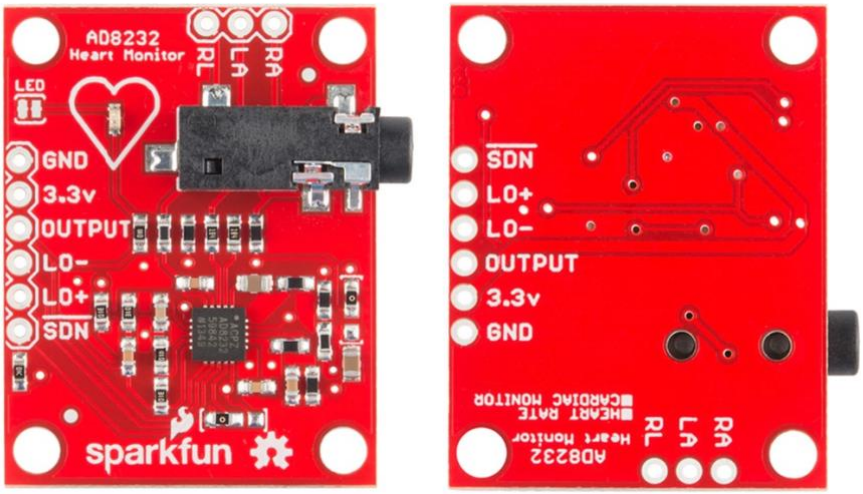


Figure 2: AD8232 heart rate monitor pin layout. Chip produced by Sparkfun [1].

Furthermore, a motor driver will be used as the third component of the bioinstrument. The chip will be powered by the MSP430 and will receive a digital input to control the motor from the microprocessor. The specific motor driver used in this instrument is also developed and produced by Sparkfun. The specific model is the TB6612FNG Dual motor driver. Figure 3 depicts the pin diagram of the motor driver used in this instrument.

Figure 3: TB6612FNG Motor Driver Pin layout

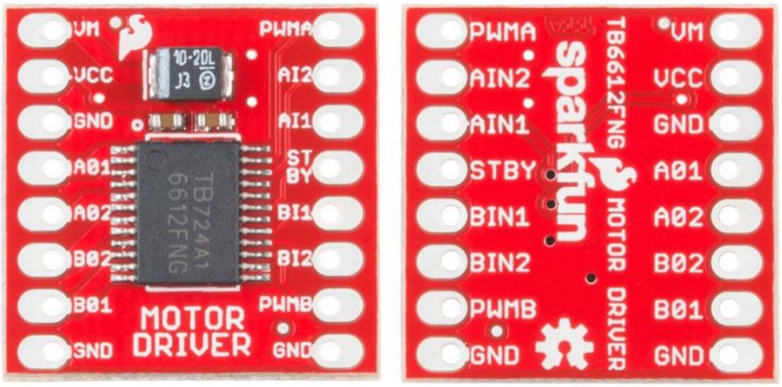


Figure 3: TB6612FNG dual motor driver pin layout. Chip produced by Sparkfun [2].

In conclusion, using the three components described above to interact with one another with the MSP430 acting as the main processor and communication center, the system will read an ECG signal from an individual using the ECG module chip and control a motor using the motor driver chip. A block diagram of the overall design is shown in Figure 4.

Figure 4: Diagram of Hardware

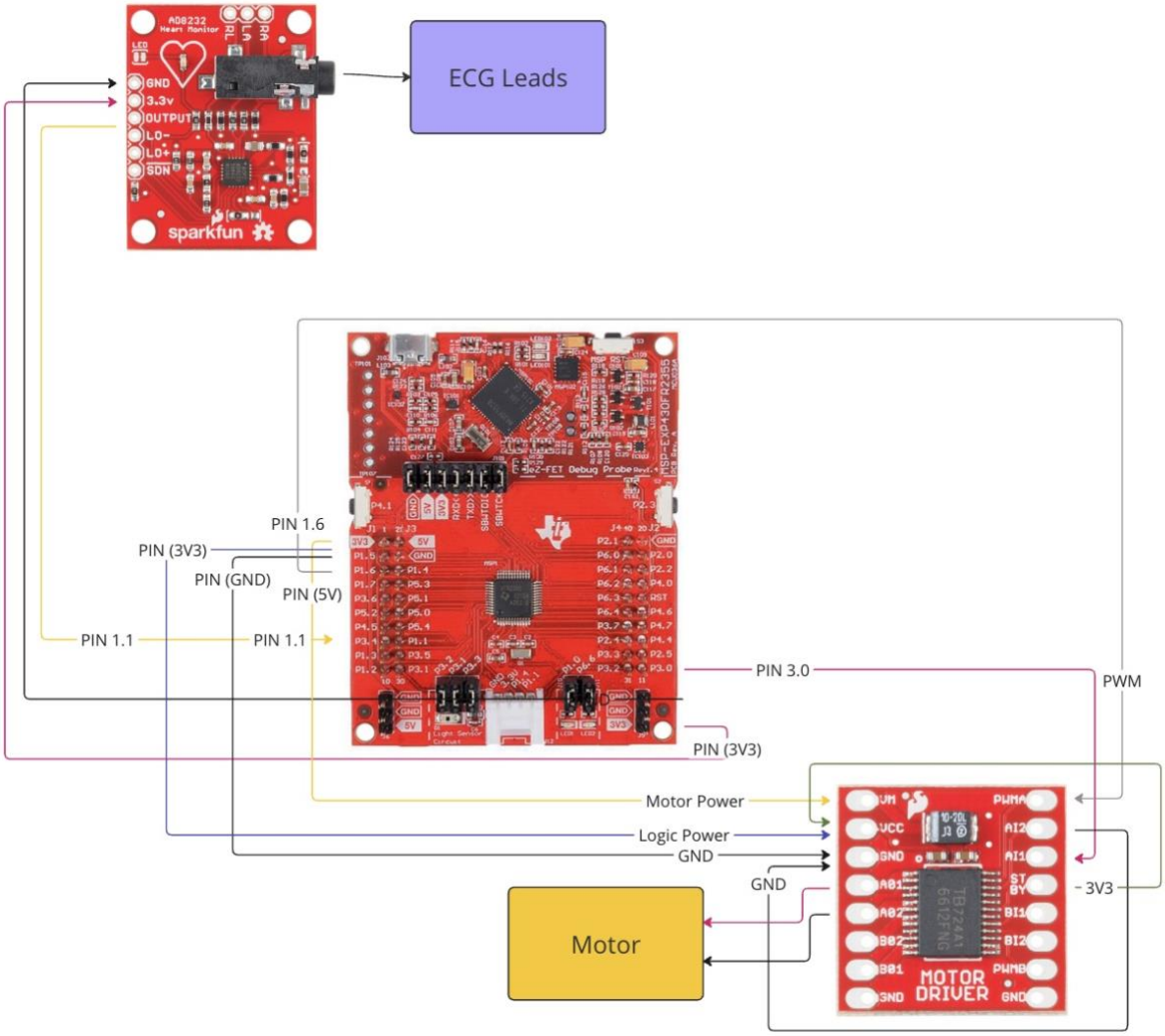


Figure 4: Block Diagram of Hardware, including MSP430, ECG module, and motor driver. Wire connections are labeled appropriately.

Component Functions:

MSP430-FR2355:

The MSP430's main hardware function is to perform as the brain of the instrument. This means that it is responsible for controlling the various components described above and labeled in Figure 4. Furthermore, it is responsible for executing the software that will be uploaded into its memory. In this instrument, the MSP430 will also be responsible for processing the analog signal produced by the heart rate monitor and converting it into a digital signal. Therefore, it will be responsible for providing a digital output signal to the motor driver chip.

TB6612FNG Dual Motor Driver:

The dual motor driver chip is responsible for controlling the motor on the bioinstrument. Moreover, the motor power and logic supply voltages are both supplied by the MSP430 in order

to allow the chip to process the input signal and supply voltage to the DC motor. Furthermore, it will receive a PWM signal from the microprocessor via PIN 1.6. The standby pin will be supplied by the 3V3 line powering the logic input. These components interacting allow for the control of the motor through the high and low state of the digital output supplied to the AI1 pin on the motor driver. The motor will be powered by the chip when the digital signal coming from the MSP430's PIN 3.0 is in the high state. Furthermore, the hardware component operates via an H-bridge. This means that the current is directed to the motor depending on how the transistors are enabled on the chip. Moreover, the AI1 and AI2 pins on the motor driver chip are responsible for changing the direction of the current going through the motor. Figure 5 depicts a schematic of an H-bridge motor driver.

Figure 5: H-Bridge Circuit Diagram

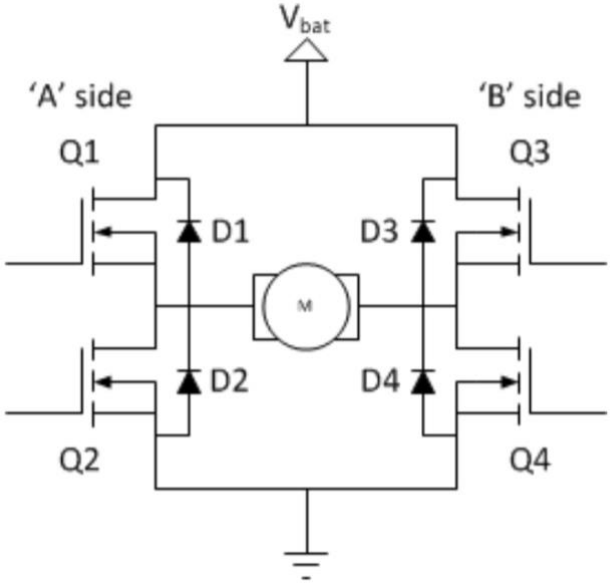


Figure 5: H-bridge diagram of motor driver circuit. Transistors are labeled by 'Q'. The motor is depicted in the center of the circuit [3].

As shown in Figure 5, there are 4 transistors on the chip in the shape of an H, thus coining its term as an H-bridge. When Q1 and Q4 are turned on, the current flows through the motor from left to right, however, when Q2 and Q3 are turned on, the current flows through the motor from right to left [3]. This format is adjusted by the high and low states being inputted into the motor driver chips by AI1 and AI2. When AI1 and AI2 are both at the low state, all transistors will be turned off. However, when AI1 is at its high state, Q1 and Q4 will be open. The opposite occurs when AI2 is switched from the high and low state for Q2 and Q3. This is how the direction of the motor is determined, by altering the states of AI1 and AI2. For the application of this instrument, only AI1 will be modified. This means that the motor will only turn in one direction. Figure 6 depicts the transistor diagram and current flow of the H-bridge depending on activation of the transistors.

Figure 6: H-Bridge Current Flow Diagram

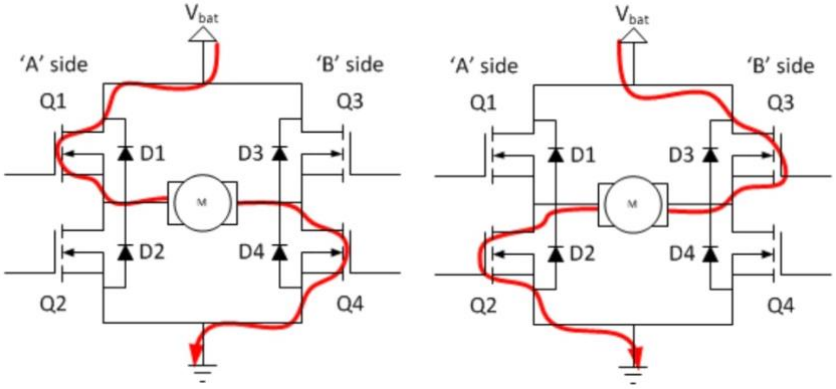


Figure 6: H-bridge diagram of current flow based on transistor configuration. Transistors are labeled by 'Q'. The motor is depicted in the center of the circuit [3].

AD8232 Heart Rate Monitor:

The purpose of the ECG module chip is to receive a cleaner signal of the patient's heart activity during monitoring. The heart rate monitor chip is responsible for producing an analog signal of the patient's ECG. The chip is powered by the MSP430 through the 3V3 supply voltage pin and is grounded by the MSP430 too. The output of the ECG module chip will be sent to the MSP430 in order to be processed and converted into a digital signal. The chip works by filtering and amplifying the signal through a series of high and low pass filters. The high pass filter is implemented in order to remove low frequency noise, this helps preserve the complexes of the signal such as the QRS peak which will be extracted from the data in data processing. Furthermore, the low pass filter included in the circuit is meant to remove extremely high frequency noise caused by muscle movement. Overall, it is designed to amplify, filter, and extract noise from the signal received, and is usually created by motion or incorrect electrode placement [2].

Input/Output Interfaces:

Table 1: Input and Output Interface of each component/chip

Chip:	Input:	Output:
MSP430	• Analog ECG signal received from the ECG module.	• Power supply to both the ECG module and motor driver • Digital GPIO pin to control motor activity
Motor Driver	• Motor and logic voltage supply • PWM for motor speed • Digital signal for motor direction and activity	• Voltage output to control motor speed and direction
ECG Module	• Patient heart rate activity	• Continuous analog signal of patients ECG heart activity

Control Mechanisms:

The motor driver uses a control mechanism for both the motor speed and direction. The activation and direction of the motor was previously described in the Component Functions section of this report. Additional controls that are important to mention about the motor driver are the PWM input and the standby input. The PWM input is responsible for controlling the speed of the motor, while the standby input controls whether the motor driver is active or in standby mode. The standby mode puts the motor in a sleep state and is put in this state when the pin is at low voltage. Standby is deactivated when the pin receives a signal in the High-voltage state, thus allowing the motor to spin. Furthermore, the PWM input receives a frequency signal in order to control the speed of the motor. A high frequency will resemble a quicker motor speed, and the signal being turned on for a greater amount of time during each period [4]. Moreover, a slower frequency will result in the signal being off for a larger portion of the signal. This is shown in Figure 7, for the relationship between duty cycle and period. A higher frequency is proportional to a higher duty cycle.

Figure 7: Duty Cycle vs Period Activation

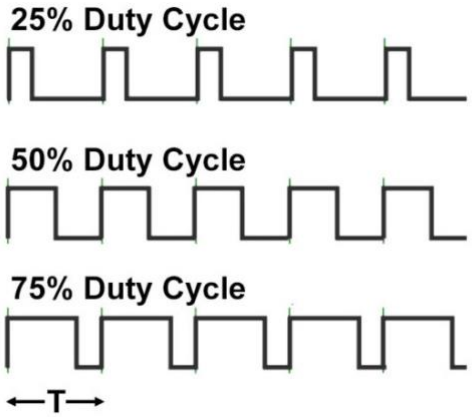


Figure 7: Duty cycle relation with signal period [4].

For this device, the MSP430 will supply a 75% duty cycle to the PWM pin on the motor driver for motor A. This will allow the motor to spin at a reasonable rate, without having the risk of running the motor at max speed or damaging it.

Software:

Software Architecture:

The MSP430 will operate under a loaded software coded in C. The software has two main functions. The first function is an ADC converter module to convert the analog signal from the ECG module into a digital signal that will be sent to the motor driver input pin. The second software module is the PWM generator for the motor rotation speed. The ADC converter will work by using an interrupt service routine, which exits the main function and determines the ADC result for each analog measurement taken.

Code Breakdown:

The compiled code within the MSP430 is shown below on the left. The function and breakdown of each line of code is shown in the textbox to the right of the code:

```
include <msp430.h>
//Create Global Variable of ADC_Result
unsigned int ADC_Result;

int main(void)
{
   WDTCTL = WDTPW | WDTHOLD;        // Stop WDT


   //Motor Control PWM Setup
   P1DIR |= BIT6;                          // P1.6 output
   P1SEL1 |= BIT6;                         // P1.6 options select

   // Disable the GPIO power-on default high-impedance mode to activate
   // previously configured port settings
   PM5CTL0 &= ~LOCKLPM5;

   TB0CCR0 = 128;                          // PWM Period/2
   TB0CCTL1 = OUTMOD_6;                    // TBCCR1 toggle/set
   TB0CCR1 = 96;                           // TBCCR1 PWM duty cycle
   TB0CTL = TBSSEL_1 | MC_3;               // ACLK, up-down mode



   // Configure GPIO for LED indicator
   P1DIR |= BIT0;                          // Set P1.0/LED to output direction
   P1OUT &= ~BIT0;                         // P1.0 LED off

   // Configure ADC A1 pin:
   P1SEL0 |= BIT1;
   P1SEL1 |= BIT1;

   //Configure Indicator PIN/Digital Output:
   P3DIR |= BIT0;                                  // Set P3.0 as output
   P3OUT &= ~BIT0;                         // Clear P3.0 (set it low)


   // Configure ADC12
   ADCCTL0 |= ADCSHT_2 | ADCON;      //ADCON, S&H=16 ADC clks



   ADCCTL1 |= ADCSHP;                      // ADCCLK = MODOSC;
                                           //sampling timer


   ADCCTL2 &= ~ADCRES;                     // clear ADCRES in ADCCTL
```

Brief Code Outline Report:

1) Creates a Global Variable called ADC_Result.

2) Initialization of the main function.

3) Stops the watchdog timer by inserting the watchdog password (WDTPW) and stops the timer from counting down (WDTHOLD).

4) Sets the PWM output pin to P1.6. This will be the pin that controls the motor's Pulse Width Modulation.

5) Disables the GPIO power-on default, so that previously configured settings are saved rather than going into a default high-impedance state.

6) Sets up the PWM signal for the motor control:
   a. Sets the PWM period to 128 ticks
   b. Toggle/set for capture Compare register for Timer B0.
   c. PWM duty cycle is set to 96 ticks or (96/128) *100 = 75%.
   d. TBSSEL_1 sets the ACCLK clock for the Timer, and M3 sets it to up-down mode.

7) For additional verification, toggle LED pin as output so that it will later be set to flash with every heartbeat.

8) Turns on the Primary and Secondary peripheral functions for Pin 1.1 in order to use the Analog to Digital Converter as a peripheral function.

9) This will set Pin 3.0 to the Digital Output for the result of the ADC converter.

10) This section of the code sets up the ADC12:
   a. ADCCTL0 is the main control register for the ADC. This line sets the sample & hold time to 16 clock ticks. Furthermore, the ADCON command sets the ADC to be turned on.
   b. ADCCTL1 is an ADC control register for additional tasks. In this register, we set the sampling timer with a preset value held by ADCSHP.
   c. ADCCTL2 is an additional ADC control register for additional commands. For my MSP430, it allows me to set the ADC resolution or number of bits used in the calculations. Here, we are clearing the resolution that is currently stored in the MSP, so it can be set later.

```
    ADCCTL2 |= ADCRES_2;                    // 12-bit conversion results

    ADCMCTL0 |= ADCINCH_1;                  // A1 ADC input select; Vref=AVCC

    ADCIE |= ADCIE0;                        // Enable ADC conv complete interrupt

    while(1)
    {
      ADCCTL0 |= ADCENC | ADCSC;            // Sampling and conversion start

      __bis_SR_register(LPM0_bits | GIE);   // LPM0, ADC_ISR will force exit
      __no_operation();                     // For debug only

      if (ADC_Result < 0x990) {             //990 (Artificial Heart)
                                            //A5A Physical Patient (2.25V) (2500)
        P1OUT &= ~BIT0;                     // Clear P1.0 LED off
        // Turn off the power source
        P3OUT &= ~BIT0;                   //P3.0 is the power control pin for motor
      }
      else {
          P1OUT |= BIT0;                    // Set P1.0 LED on
          // Turn on the power source
          P3OUT |= BIT0;                    //P3.0 is the power control pin
          __delay_cycles(50000);
      }

      __delay_cycles(10);                   //5000 (Testing)

    }
}

// ADC interrupt service routine
#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__
#pragma vector=ADC_VECTOR
__interrupt void ADC_ISR(void)
#elif defined(__GNUC__)
```

d. After clearing the register, we are setting the ADC resolution to 12-bit, by setting the control register to ADCRES_2, which holds the bit pattern for 12-bit conversion.

e. This line of code sets the ADC module to select input channel #1. Since we are working with P1.1, A1 is the acceptable input channel that works with this model of MSP430 as shown in Figure 1. Furthermore, it also sets the internal voltage reference to the Analog VCC value. In this case, it is the supply voltage to the microcontroller.

f. This line of code enables the ADC conversion complete interrupt.

11) This loop is the actual process of the ADC taking place and computing the output. Furthermore, depending on the ADC_Result, the motor input and LED light will be toggled.
   a. This first line changes the properties of the ADC control register by enabling it (ADCENC) and then starting the conversion (ADCSC). By using |=, we are setting these commands for the control register.
   b. This line of code calls to the status register and forces an ADC complete interrupt (using a global interrupt with GIE). It will utilize the ADC_Result global variable, which will be compared in the next line of code.
   c. This relation compares the ADC_Result from the global interrupt to a threshold voltage. In this code, I am comparing it to the hex value (0x990), which is ~2450 in binary. This corresponds to about 2.45V. When the result is below 2.45V, the P1.1 is kept in the low state (LED off), however, when the result is greater than 2.45V, the port is in the high state (LED on).
      i. This threshold is changed based on whether a patient is using the device, or an artificial heart is connected.
      ii. Delay in else statement keeps the digital high state on for longer, allowing the motor to spin longer.

12) The following code is an interrupt service for confirming the ADC_Result. It uses a switch case scenario where the ADC_Result is set to ADC memory buffer 0 (First memory location) as long as it has properly been converted. The case statements check various interrupt controls in the ADC interrupt service to make sure that a digital value was determined (case ADCIV_ADCIFG).

```
void __attribute__ ((interrupt(ADC_VECTOR))) ADC_ISR (void)
#else
#error Compiler not supported!
#endif
{
  switch(__even_in_range(ADCIV,ADCIV_ADCIFG))
  {
    case ADCIV_NONE:
      break;
    case ADCIV_ADCOVIFG:
      break;
    case ADCIV_ADCTOVIFG:
      break;
    case ADCIV_ADCHIIFG:
      break;
    case ADCIV_ADCLOIFG:
      break;
    case ADCIV_ADCINIFG:
      break;
    case ADCIV_ADCIFG:
      ADC_Result = ADCMEM0;
      __bic_SR_register_on_exit(LPM0_bits);   // Clear CPUOFF bit from LPM0
      break;
    default:
      break;
  }
}
```

13) The other interrupt calls are defined below:
    a. ADCIV_NONE: No interrupt.
    b. ADCIV_ADCOVIFG: overflow interrupt
    c. ADCIV_ADCTOVIFG: time overflow
    d. ADCIV_ADCHIIFG: ADC conversion result higher threshold.
    e. ADCIV_ADCLOIFG: ADC conversion result lower threshold.

Data Structures:

One data structure used by the microcontroller is an interrupt service routine, which is responsible for exiting the main function and performing a routine service calculation to continue the code. This service routine was provided by one of the sample files within the TI code library (msp430fr235x_adc12_01.c) and modified to be used within this project. Within this code, the interrupt service acts on the analog to digital converter (ADC). The analog to digital converter works by comparing the received analog signal to an ideal transfer characteristic, and depending on whether it is over or under the line, it will convert the analog signal to a digital representation using a certain resolution. As shown in the code, the ADC resolution used within this device was 12-bit. Therefore, the digital resolution was very accurate when converted to a digital value, however, it can be noted that this was not necessary as the direct digital value was only being used to compare. By converting the signal to a digital value, it allowed for an easier relation to a predetermined threshold to compare whether or not to turn the GPIO motor controller pin to a high or low state. Figure 8 depicts an example of how an ADC works using a 3-bit resolution.

Figure 8: Analog to Digital Converter Diagram

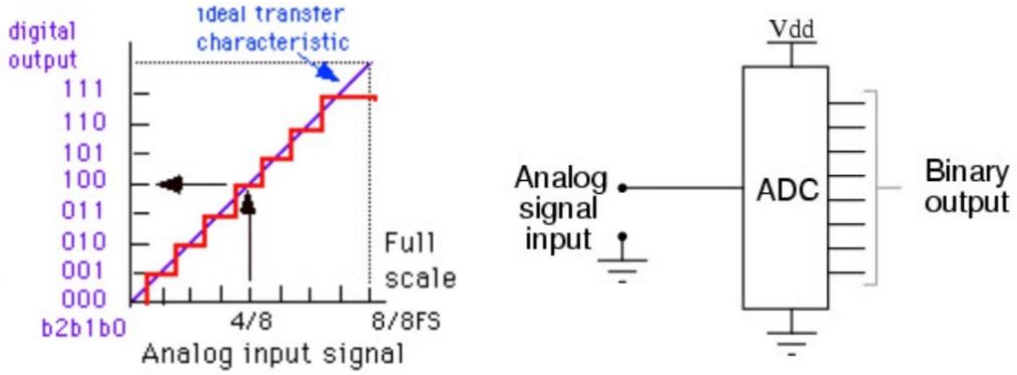


Figure 8: Analog to Digital Conversion Process. Example uses 3-bit; however, code implementation uses 12-bit resolution [5].

Furthermore, the interrupt service routine saves the value of each calculation by using a global variable. Global variables are data structures in C that exist outside of the main function, and therefore can be accessed by exterior functions. In the case of this software, this allows us to access the ADC_Result value within the interrupt service which exists outside of the main function. This global variable is created by calling the following code: unsigned int ADC_Result.

Control Flow:

The flow of control for this system mainly takes part in the while loop at the end of the code. As previously stated, within this while loop, there consists of a continuous call to the ADC interrupt service routine. This service routine halts the code, so that it can exit the main program and calculate the ADC result. As shown in the code explanation above, there are also a series of case statements which verify that the digital result was calculated successfully without throwing an error. After each ADC conversion, the result is compared to a threshold determined through calibration techniques. If the ADC result is greater than the threshold, then the digital port connected to the motor driver is switched to its high state allowing for the motor to be turned on. In contrast, if it is less that the ADC result, then the digital motor control pin is turned off by setting it to its low voltage state. In conclusion, the software responds to the different analog inputs and produces a varying digital response based on the input. This varying digital response triggers the motor control of the system.

Summary:

In conclusion, the hardware interacts with the software on the MSP430 in order to convert the analog signal received from the ECG module. The converted analog signal is outputted as a digital signal to the motor driver which turns the motor on and off depending on whether the analog signal is above a certain threshold. For this project, this occurs when the ECG activity reaches the R peak of the heart cycle and will be explained further in the device calibration below.

Device Calibration:

Introduction:

Device calibration is extremely important for instrumentation. Calibrating a device ensures that it works repetitively and in the same way with each operation. Furthermore, calibration techniques are implemented to ensure that the input signal to a device is clean enough for computation by reducing signal noise. For this device, the ECG module acts as a calibration tool to clean up and clarify the analog signal received by the MSP430.

Calibration Process Overview:

Physical Measurement to Signal Conversion:

As previously stated, the ECG module shown in Figure 2 is primarily responsible for much of the device input signal calibration. The ECG module is responsible for converting the physical heart activity recorded by the device into an analog signal that is sent as a continuous electrical signal to be processed by the microcontroller. The signals are recorded by placing an electrode on the right wrist and left wrist, as well as a grounding electrode to another point on the body. Figure 9 depicts the proper placement points of the ECG electrodes in order to obtain accurate physical heart recordings.

Figure 9: Proper Lead Placement for ECG

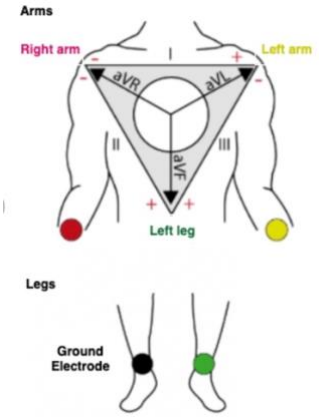


Figure 9: Proper lead placement for ECG. Ground electrode can be placed anywhere on the lower body [6].

Signal Conditioning:

The ECG module conditions the analog signal that is interpreted by the microprocessor through a series of high and low pass filters. In general, high-pass filters allow high-frequency signals to pass while attenuating low-frequency signals, while low-pass filters allow low-frequency signals to pass while attenuating high-frequency signals. This allows the device to allow a certain frequency input in while removing outlying noise factors. As a result, the filters allow only the wanted portions of the heart activity while removing small muscle movements and other noise applications from the surroundings. Furthermore, the ECG module also contains an amplification component which creates a higher reference potential that is sent to the microcontroller. This is very useful in the application of this instrument, as it brings all electrical

signal activity above 0V. Since most microcontrollers cannot interpret bipolar signals, without the amplification and reference offset created by the ECG module, the microcontroller would not be able to understand the preprocessed signal. Figure 10 depicts the analog signal sent to the microcontroller post processing.

Figure 10: Analog Heart Activity – Post Processing by ECG Module

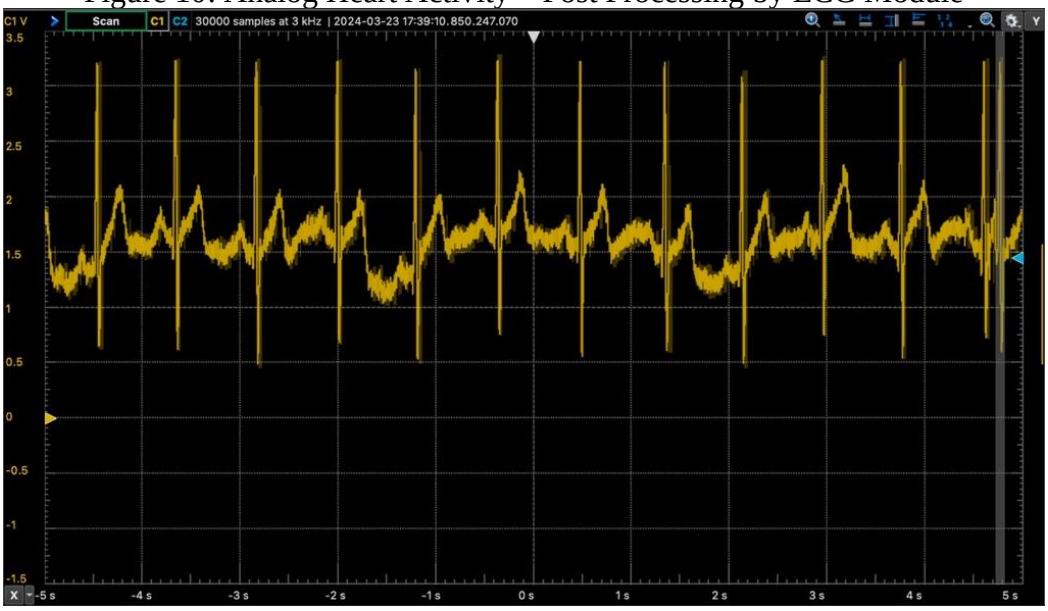


Figure 10: ECG example post-processing by ECG module chip. Voltage scale is shown on the left.

Microcontroller Signal Processing:

Signal Input to Microcontroller:

The input that the MSP430 receives will be analog, therefore, is inputted into PIN 1.1, or A1, which can handle analog input. The peripheral functions of this pin are also enabled in the code to allow for analog to digital conversion by the microcontroller. It is important to note that the operating voltage for the built in ADC module is between 0V and the supply voltage Vcc. Since the microcontroller is powered by a laptop, the supply voltage is 5V. As shown in Figure 10, the analog input ranges between 0V and 3.5V, therefore, there will be no issue for the built in ADC module.

Post-Processing:

Data Analysis:

As previously described, in the data structures section of the software analysis, the signal input to the microcontroller is analyzed by the built in ADC unit on the microcontroller. This signal is converted to a digital reference at every 16 clock ticks. Therefore, the global variable will be replaced every 16 clock ticks as designated by the sample and hold variable. This allows for comparison against a threshold value in order to toggle the motor controller. Figure 11 depicts the motor controller pin being set to the high state at a specific threshold value of 2.2V for the analog input.

Figure 11: Analog Heart Activity – Post Processing & GPIO Motor Activation

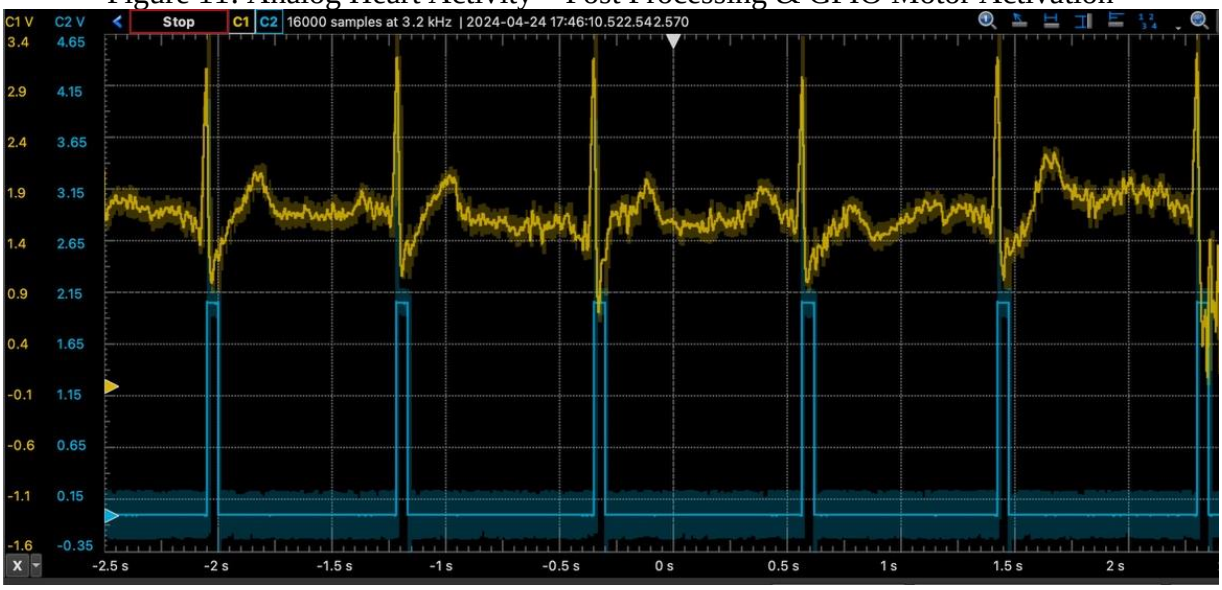


Figure 11: Analog post-processed input signal to MSP430, and digital output GPIO pin signal after comparison.

Summary:

This section covers how the ECG module converts physical heart activity into an analog signal for processing by the microcontroller, by using a series of amplifiers, high pass, and low pass filters to remove noise from the signal. Furthermore, the ECG module conditions the analog signal to be within 0V and 5V to ensure that the microcontroller can interpret the signal effectively. Moreover, the analog signal is inputted into the microcontroller and converted to a digital value using the built-in ADC module. This allows the microcontroller to analyze and compare the converted input against a threshold value to toggle the motor controller between its high and low voltage state. As a result, device calibration is important in maintaining measurement accuracy and reliability. Excess noise can create issues where the motor controller would be activated more often than it should be.

Applicable Physiology:

Introduction:

The purpose of this section of the report is to introduce the applicable usages of this instrument in regard to physiology and medical instrumentation. This instrument uses an ECG module; therefore, it is relevant to the application of the cardiovascular system and heart rate function. As a result, it is important to understand the cycles of the heart and any irregularities for the proper operation of this device.

Physiological Connections

Relevant Body System:

Since this device operates under a normal ECG cycle, it is important to discuss the physiology of the heart. As shown in Figure 12, a normal sinus rhythm ECG is depicted of an average adult.

Figure 12: Normal Adult Heart Cycle

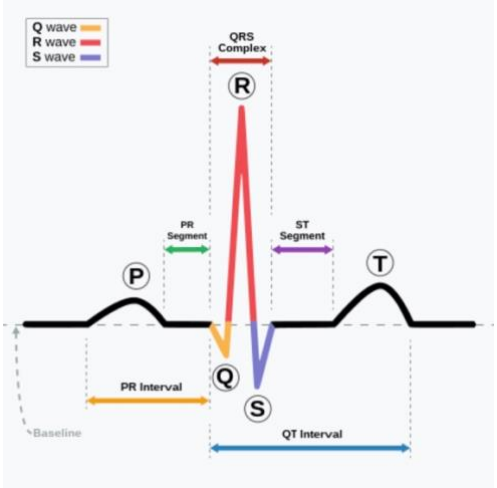


Figure 12: Normal adult sinus rhythm ECG wave [7].

As shown in Figure 12, the typical rhythm of the heart can be broken down into three sections. The first section is the P-wave, the second the QRS-complex, and the third the T-wave. The P-wave is the first electrical wave of the heart, and is caused by atrial depolarization, and is relatively smooth, depicting the smooth contraction of the atria. Furthermore, the QRS-complex is caused by ventricular depolarization, which is the quick contraction of the ventricles to push blood out of the heart. This is followed by the T-wave, and is represented by the ventricle repolarization, or when the ventricles recover to their resting state [6].

Key Parameters/Physiological Signal Isolation:

This device will focus on the **QRS complex** of the heart cycle. Therefore, it is looking at isolating the ventricular contraction from the sinus rhythm when performing data analysis and variable comparisons. This is because the device aims to set a trigger for each contraction of the heart, therefore, it is best to isolate the heart cycle at the R peak as it has the highest electro potential during the heart's cycle. By focusing on the highest point, it removes chances of error in detecting more than one cyclic contraction in a single heartbeat. For example, if the threshold is set too low, then it may trigger additional pulses during the T-wave, as it has the second highest electro potential. This potential issue is shown in Figure 13.

Figure 13: Multiple Trigger Potential – Failed R Peak Isolation

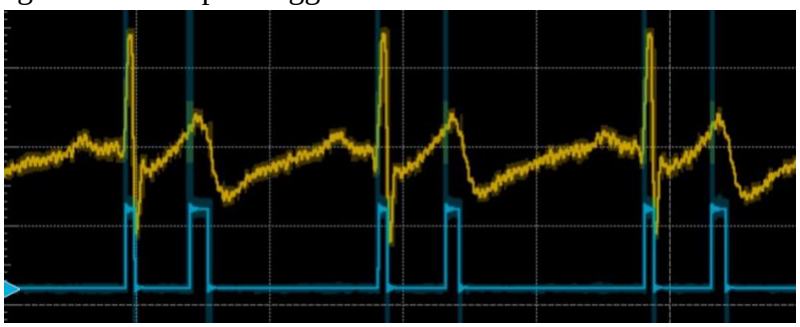


Figure 13: Multiple trigger error for failed QRS isolation.

Abnormal Physiology:

The issue described in Figure 13 can also be a result of abnormal physiological conditions of the heart. A heart condition called hyperkalemia can result in abnormally high repolarization of the ventricles. This could further lead to the issue present in Figure 13, where the microcontroller signals that more than one heartbeat has taken place in a single heart cycle. Figure 14 depicts a normal vs hyperkalemic sinus rhythm.

Figure 14: Hyperkalemia – Large T Wave

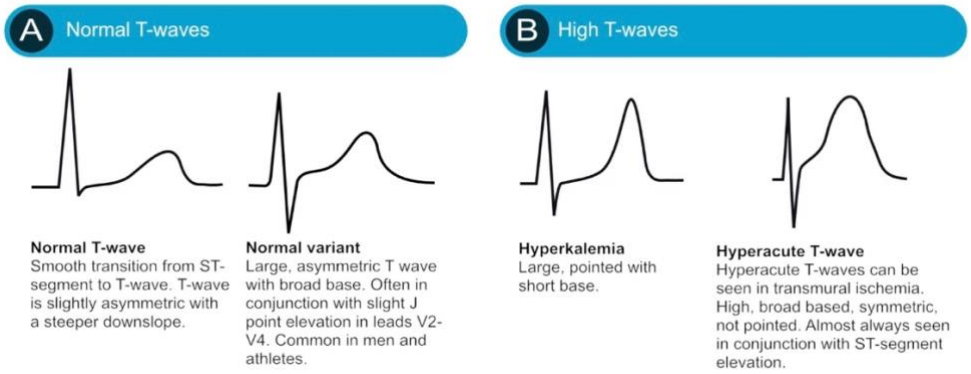


Figure 14: Hyperkalemia example showing high T wave electro potential [8].

As shown in Figure 14, the T wave in individuals with hyperkalemia may induce additional activations of the motor controller as the threshold of their electro potential during their T wave (ventricular repolarization) may surpass as it does during their QRS complex. As a result, the induced representation of every heartbeat that the instrument performs will not be accurate.

Clinical Relevance:

While this device does not have specific clinical relevance, it can be used to show that it has clinical applications such as depicting heart rate and actively changing systemic behavior as a result of such monitorization. This device could be compared to a simple heart monitor where it depicts the ECG of a patient and performs data analysis on their real time heart activity.

Summary:

The instrument's application with physiology and medical instrumentation focused on its utilization of the ECG module, which aligns with the cardiovascular system and heart rate functions. Understanding the heart's cycle and any irregularities is crucial for the device's proper operation. The device focuses on isolating the QRS complex for digital comparison and analysis, aiming to trigger a motor and LED with each heart contraction at its highest electro potential point (QRS complex). Abnormalities, such as hyperkalemia-induced high T waves can be a potential issue with operating this device as it can isolate more than one cycle in a given heartbeat. While the device lacks specific clinical relevance, it demonstrates the potential applications akin to heart monitors, offering real-time ECG depiction and analysis of heart activity.

Works Cited:

[1] "Sparkfun Single Lead Heart Rate Monitor - AD8232." *SEN-12650 - SparkFun Electronics*, www.sparkfun.com/products/12650. Accessed 24 Apr. 2024.

[2] "Sparkfun Motor Driver - Dual TB6612FNG (1A)." *ROB-14451 - SparkFun Electronics*, www.sparkfun.com/products/14451. Accessed 24 Apr. 2024.

[3] Tantos, Andras. "H Bridges - The Basics." *Modular Circuits*, 2011, www.modularcircuits.com/blog/articles/h-bridge-secrets/h-bridges-the-basics/.

[4] "Led Dimming Using the Pulse Option and PWM." *NIGHTSEA*, 6 Mar. 2023, nightsea.com/articles/pwm-led-dimming/.

[5] Agarwal, Tarun. "Analog to Digital Converter : Block Diagram, Types & Its Applications." *ElProCus*, 29 Jan. 2021, www.elprocus.com/analog-to-digital-converter/.

[6] Ayano, Yehualashet Megersa, et al. "Interpretable Machine Learning Techniques in ECG-Based Heart Disease Classification: A Systematic Review." *Diagnostics (Basel, Switzerland)*, U.S. National Library of Medicine, 29 Dec. 2022, www.ncbi.nlm.nih.gov/pmc/articles/PMC9818170/.

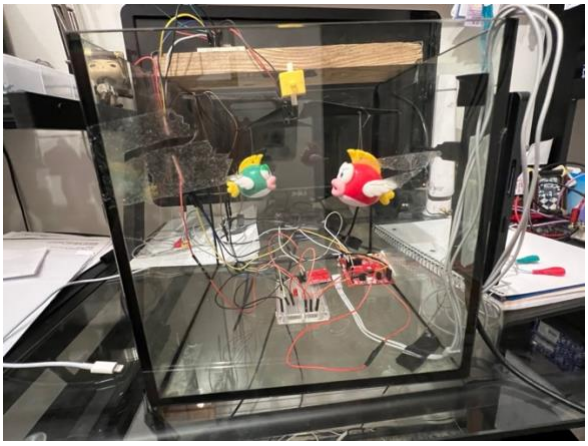
[7] "QRS Complex." *Wikipedia*, Wikimedia Foundation, 8 Apr. 2024, en.wikipedia.org/wiki/QRS_complex.

[8] "ECG Interpretation: Characteristics of the Normal ECG (P-Wave, QRS Complex, St Segment, T-Wave)." *Cardiovascular Education*, 25 June 2023, ecgwaves.com/topic/ecg-normal-p-wave-qrs-complex-st-segment-t-wave-j-point/.

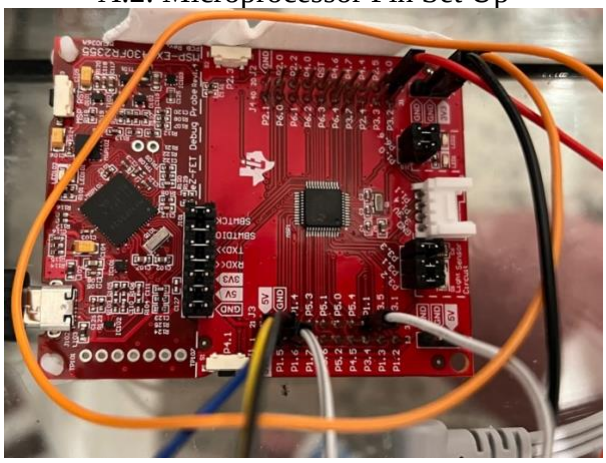
Appendix:

Additional Photos of Device:

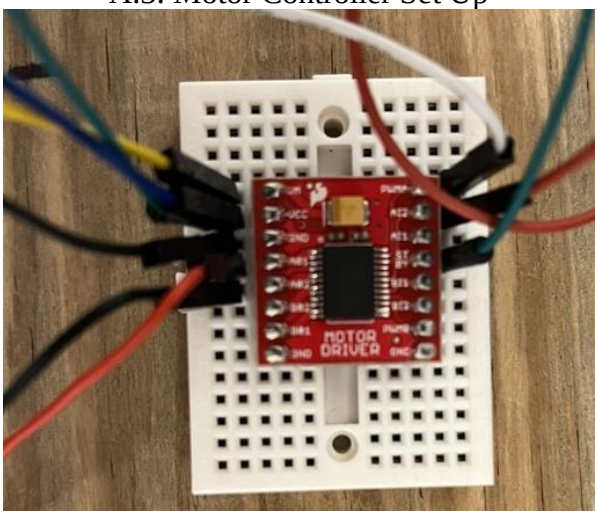
A.1: Assembled Device



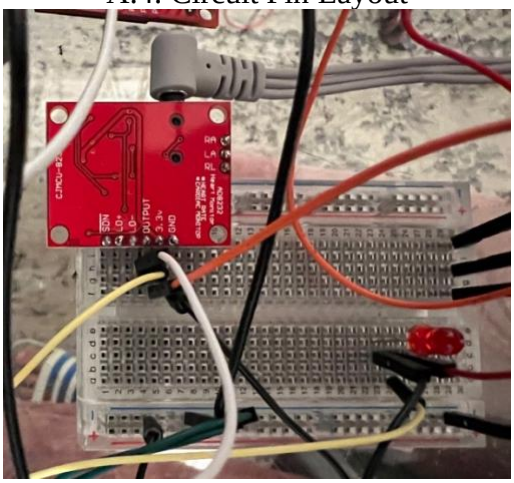
A.2: Microprocessor Pin Set Up



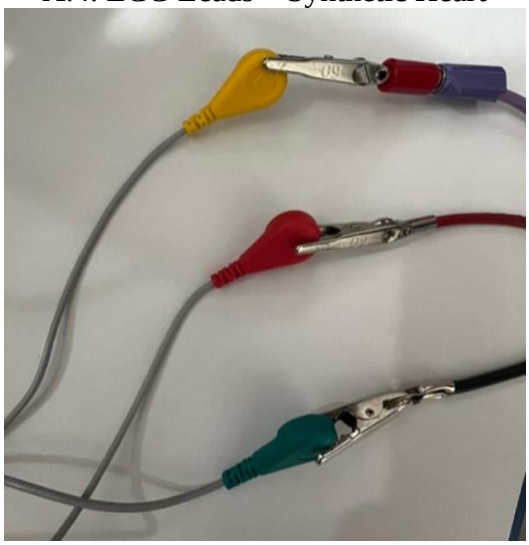
A.3: Motor Controller Set Up



A.4: Circuit Pin Layout



A.4: ECG Leads – Synthetic Heart



A.5: Synthetic Heart Simulator