Name: Saurav Agrawal
Subject: Design of Mechatronic Systems (MEAM 5100)
17 October 2023

3.1.2: We need a way of reading ADC channels continuously. Create two subroutines: one for setting up an arbitrary ADC port (so the user can specify what port they would like to set up), and one for reading the setup ADC port. Try to make your subroutine as clean and efficient (short and fast) yet also general for any ADC port. So, it is a good idea to have the port (or ports) you are initializing as a parameter you pass to your subroutine. We will post the routine we like best as an example for the rest of the class. One goal of this part is to get practice writing a subroutine that is general so others can use it in arbitrary projects. Use the set and clear macros not the bit manipulation directly. Optional: write the routines (or add more) that will allow the efficient reading of multiple ADC ports. Submit your well commented code (we will have a separate submission for code). If you use other resources like ChatGPT include your prompt, the output and explain what each part does and how you modified it for this lab.

Answer: I have created a subroutine to set up a port in the Itsy Bitsy and then to read the channel. The functions are called '**void adc_ch(uint8_t adcch)**' and '**int adc_read()**'. To read arbitrary channels I have used if statements since there are only 9 channels that can be used in the Itsy Bitsy for our code. The Sub routines follow the 10 steps to define ADC that were taught in the lecture.

1. set the voltage reference
2. set the ADC clock prescaler
3. disable ADC pin digital input
4. optional: (set up interrupts and triggering) **Did not set the interrupt in my code**
5. select the desired analog input
6. enable conversions
7. start the conversion process
8. wait for conversion to finish
9. read the result
10. clear the conversion flag

Code : **Since it is a big file I am attaching the zip code of the file.**

I have used ChatGPT to get the layout of the code like function defining and calling.

I had used the prompt : Create subroutines for setting up and reading ADC channels.
This prompt gave me a basic structure to set up the code and then I used the 10 points mentioned above to write the code. I have used set and clear to write the code:

3.1.3. Interface the ItsyBitsy with two analog sensors (such as potentiometers) so their positions can be displayed, printing to the USB serial port. Build the sensing side of your Waldo and submit a video of it moving with the position of the joints being moved and the angle being displayed in a serial window on a computer. Submit a link to a video (i.e., youtube, etc.) of your device and the serial window showing the ItsyBitsy values reading motion, include drawings, and any code used. Provide a short discussion of the sensitivity of your device (the number of ADC counts over the full range of motion, linearity and apparent noise sources.)

Answer: The link to the video of the sensing side is attached below:
https://www.youtube.com/playlist?list=PLI1MCD4tyefPC4wbL-LwrJVAicPKKOZv6

The playlist has 2 videos 1 showing the ADC reading and the other showing the angle reading using the angle ranging from 0 to 180 degrees.There are a total of 1024 steps between 0-180 degrees in the ADC. Each step is 0.1757 degrees.To analyze noise sources, I observed fluctuations in sensor readings when the potentiometers are stationary. It was quite stable as we can see in the above video also since I had soldered the wire and used a heat shrink to completely seal the loose points and also the bread board was kept in a stable place which resulted in a very low noise. As we can see in the video I have moved the potentiometer from 0 to 1024 ADC readings. To check the full range of motion that the potentiometer provides. And as the potentiometer rotates the ADC values change linearly(decreases or increase linearly) we can use the readings from the ADC which is received in the videos and plot it with respect to time to get a graph which is almost linear.