**High Level**

**Basic Operation:** The program uses TIMER1 interrupts to initiate each ADC conversion manually, which allows for better control over the sample rate.

**Setup Pins**

* Set PD1 and PD2 as output pins.
* Set PA0 and PA1 as input pins.

**Setup Timer1**

* Initialize TIMER1 counter TCNT1 to 0x00.
* Set TIMER1 prescaler to 1 so TIMER1 = 16 MHz.
  + Frequency of timer interrupt = system clock frequency/(prescaler \* compare match register value + 1)
* Timer operates in phase correct, PWM mode. TOP value is stored in ICR1, since OCR1A/B are used to control PWM output pins.
  + Set WGM13 to 1, WGM12 to 0, WGM11 to 1, WGM10 to 0.
* Maximize PWM frequency.
  + PWM frequency = timer
  + Write 0xFF to ICR1L.
  + Write 0xFF, shifted 8 bits to ICR1H.

**Setup ADC**

* Use AVCC as reference voltage -> Write 1 to ADMUX.REFS
* Left aligned for 8 bit resolution: write 1 to ADMUX.ADLAR
* Prescaler = 128 for 16 MHz clock
  + ADC operates from 50 KHz to 200 KHz to get maximum resolution
  + 16 MHz/128 = 125 KHz ADC frequency
  + Write 0x111 to ADCSRA.ADPS
* Initialize ADMUX.MUX to channel 0 (pin ADC0)
  + This will change periodically to channel 1 in order to read the potentiometer.
* **IF FREE RUNNING MODE:**
  + Enable ADC conversion complete interrupt.
    - Set ADCSRA.ADIE to 1.
  + Enable auto trigger.
    - Write 1 to ADCSRA.ADATE
* Enable ADC.
  + Set power reduction ADC bit (PRR.PRADC) to 0
  + Write 1 to ADCSR.ADEN.
    - Note: Takes 25 ADC clock cycles to initialize first conversion. After that, further conversions take 13 ADC clock cycles (13.5 for auto triggered conversions).
* Enable global interrupts.
* Start a single conversion.
  + Set start conversion bit to 1 in ADCSRA.ADCS.

**Change Channels Function**

* If ADMUX.MUX = 0 (ADC0 currently active)
  + Set ADMUX.MUX to 1.
* If ADMUX.MUX = 1
  + Set ADMUX.MUX to 0.

**Interrupt Service: TIMER1**

* Increment counter.
* If counter = n:
  + Switch ADC channels.
  + Reset counter = 0.
* If counter = 0:
  + Switch ADC channels
* Initiate ADC conversion.

**Interrupt Service: ADC Complete**

* **POTENTIOMETER LOGIC:** If counter = n (n to be found experimentally)
  + Read ADCH into POT. Leave ADCin unchanged.
  + Call change channels function again.
  + Reset counter = 0.
* Map upper and lower threshholds based on potentiometer value.
* Map input values to amplified equivalents.
* If (ADC value >= upper threshold)
  + Set output value = upper threshold
* If (ADC value <= lower threshold)
  + Set output value = lower threshold
* **POTENTIOMETER LOGIC:** If (++counter=n)
  + **Call change channels function** so next ADC interrupt will be for channel 1.
* Convert ADC\_high and ADC\_low into 16 bit value.
  + Write combined value to OCR1A/B.

<https://onlinedocs.microchip.com/pr/GUID-F670183D-C025-46C7-B7F5-9BAD389BF43F-en-US-3/index.html?GUID-36865EC3-4E82-41FA-B7DA-58E36435CDE9>