```c
/*
 * FileName: main.c
 * Version: 1
 *
 * Created: 10/11/2022 5:36:00 PM
 * Author: Ethan Zeronik
 *
 * Operations: set the button IO
 *
 * Hardware:
 *   Atmega2560          micro controller
 *   PORTA.0             mode switch
 *   PORTA.4             start pushbutton
 *   PORTA.5             stop pushbutton
 */

/* NOTE: Includes */
#include <avr/io.h>

#include "Debugger.h"

/* NOTE: Custom Macros */
// TODO: None

/* NOTE: Global Variables */
// TODO: None

/* NOTE: Function prototypes */
// inits IO ports
void IO_init(void);

/* NOTE: Application implementation */
// the main loop of the function, provided to us
int main(void)
{
    IO_init();

    initDebug();

    while(1)
    {
    }
}

/* NOTE: Function implementations */
void IO_init(void)
{
    // set port A as all inputs
    DDRA  = 0x00;
    // turn all pullup resisistors
    PORTA = 0xFF;
}
```

```c
/*
 * FileName: main.c
 * Version: 1
 *
 * Created: 10/18/2022 8:38:44 PM
 * Author: Ethan Zeronik
 *
 * Operations: Blink and LED every 500ms
 *
 * Hardware:
 *   Atmega2560          micro controller
 *   PORTB.7             LED13 active high
 */

/* NOTE: Includes */
#include <avr/io.h>

#include "Delay.h"
#include "Debugger.h"

/* NOTE: Custom Macros */
// TODO: None

/* NOTE: Global Variables */
// TODO: None

/* NOTE: Function prototypes */
// inits IO ports
void IO_init(void);

/* NOTE: Application implementation */
// the main loop of the function, provided to us
int main(void)
{
    DLY_init();

    IO_init();

    initDebug();

    while(1)
    {
        DLY_ms(500);

        PORTB = ~(0x80 & PORTB);
    }
}

/* NOTE: Function implementations */
void IO_init(void)
{
    // set port B.7 as an output
    DDRB  = 0x80;
    PORTB = 0x00;
}

```

```c
/*
 * FileName: main.c
 * Version: 1
 *
 * Created: 10/18/2022 8:55:13 PM
 * Author: Ethan Zeronik
 *
 * Operations: make a basic PWM controller
 *
 * Hardware:
 *   Atmega2560          micro controller
 *   PORTA.4             start pushbutton
 *   PORTA.5             stop pushbutton
 *   PORTB.5             PWM output
 */

/* NOTE: Includes */
#include <avr/io.h>

#include "PulseWidthModulation.h"
#include "Delay.h"
#include "Debugger.h"

/* NOTE: Custom Macros */
// TODO: None

/* NOTE: Global Variables */
// TODO: None

/* NOTE: Function prototypes */
// inits IO ports
void IO_init(void);
// makes a ramp
void rampUpDelayWithSteps(double start, double end, uint16_t duration, uint8_t stepCount);

/* NOTE: Application implementation */
// the main loop of the function, provided to us
int main(void)
{
    IO_init();
    PWM_init();
    DLY_init();

    initDebug();

    while(1)
    {
        while((PINA & 0x10) == 0)
        {
            // do nothing
        }

        PWM_enable();

        rampUpDelayWithSteps(.1, .5, 8000, 5);

        while((PINA & 0x20) == 0)
```

```
        {
            // do nothing
        }

        PWM_disable();
    }
}

/* NOTE: Function implementations */
void IO_init(void)
{
    // set port B.7 as an output
    DDRA  = ~0x30;
    PORTA = 0x30;
}

void rampUpDelayWithSteps(double start, double end, uint16_t duration, uint8_t stepCount)
{
    double stepIncrement = (end - start) / (stepCount - 1);

    for(size_t i = 0; i < stepCount; i++)
    {
        DLY_ms(duration / stepCount);
        PWM_dutyCycle((stepIncrement * i) + start);
    }
}
```

```c
/*
 * FileName: main.c
 * Version: 1
 *
 * Created: 10/19/2022 12:21:39 PM
 * Author: Ethan Zeronik
 *
 * Operations: ADC to PWM
 *
 * Hardware:
 *   Atmega2560          micro controller
 *   PORTB.5             PWM output
 *   PORTB.7             LED13 active high
 *   PORTA.0             mode switch
 *   PORTA.4             start pushbutton
 *   PORTA.5             stop pushbutton
 *   PORTF.0             pot in
 */

/* NOTE: Includes */
#include <avr/io.h>

#include "PulseWidthModulation.h"
#include "AnalogToDigital.h"
#include "Delay.h"
#include "Debugger.h"

/* NOTE: Custom Macros */
// TODO: None

/* NOTE: Global Variables */
// TODO: None

/* NOTE: Function prototypes */
// inits IO ports
void IO_init(void);

/* NOTE: Application implementation */
// the main loop of the function, provided to us
int main(void)
{
    IO_init();
    PWM_init();
    DLY_init();
    ADC_init();

    initDebug();

    while(1)
    {
        if(PINA & 0x01)
        {
            PWM_enable();

            // set pwm to pot adc
            PWM_dutyCycle(ADC_getTenBitValue(0x00));
        }
```

```
58          else
59          {
60              PWM_disable();
61
62              // toggle led every 500 ms
63              DLY_ms(500);
64              PORTB = ~PORTB & 0x80;
65          }
66      }
67  }
68
69  /* NOTE: Function implementations */
70  void IO_init(void)
71  {
72      // set port B.7 as an output
73      DDRB  = 0x80;
74      PORTB = 0x00;
75
76      // set port A.0 as an input
77      DDRA  = 0x00;
78      PORTA = 0x01;
79
80      // set port F.0 as an input
81      DDRF  = 0x00;
82      PORTF = 0x00;
83  }
84
```

```c
/*
 * FileName: Delay.h
 * Version: 1
 *
 * Created: 10/18/2022 7:26 PM
 * Author: Ethan Zeronik
 *
 * Operations: header for the delay submodule
 */

#ifndef Delay_h_INCLUDED
#define Delay_h_INCLUDED

#if defined(__cplusplus)
extern "C" {
#endif

#pragma message("WARNING: this module uses Timer 0 for delays")

#include "stdio.h"

#define F_CPU 16000000UL

/* NOTE: Custom Types */
// TODO: None

/* NOTE: Function prototypes */
// init registers for delay
void DLY_init(void);
// delay for an amount of ms
void DLY_ms(double ms);

#if defined(__cplusplus)
} /* extern "C" */
#endif

#endif // Delay_h_INCLUDED
```

```
60          OCR0A = (((1 / 1000.0) * F_CPU) / 1024);
61
62          for(size_t i = 0; i < ms; i++)
63          {
64              // prescalar of 1024
65              TCCR0B = 0x05;
66
67              while((TIFR0 & (1 << OCF0A)) == 0)
68              {
69                  // do nothing
70              }
71
72              // stop the timer
73              TCCR0B = 0x00;
74              // clear the overflow flag
75              TIFR0 |= (1 << OCF0A);
76              TCNT0 = 0;
77          }
78
79          OCR0A = 0;
```

```c
/*
 * FileName: PulseWidthModulation.h
 * Version: 1
 *
 * Created: 10/18/2022 10:14 PM
 * Author: Ethan Zeronik
 *
 * Operations: header for the pwm submodule
 */

#ifndef PulseWidthModulation_h_INCLUDED
#define PulseWidthModulation_h_INCLUDED

#if defined(__cplusplus)
extern "C" {
#endif

#pragma message("WARNING: this module uses Timer 1 for PWM")

#include "stdio.h"

#define F_CPU 16000000UL

/* NOTE: Custom Types */
// TODO: None

/* NOTE: Function prototypes */
// init registers for PWM
void PWM_init(void);
// makes a pwm for a given duty cycle
void PWM_dutyCycle(double percent);
// enable pwm output
void PWM_enable(void);
// disable pwm output
void PWM_disable(void);

#if defined(__cplusplus)
} /* extern "C" */
#endif

#endif // PulseWidthModulation_h_INCLUDED
```

```c
/*
 * FileName: PulseWidthModulation.c
 * Version: 1
 *
 * Created: 10/18/2022 10:14 PM
 * Author: Ethan Zeronik
 *
 * Operations: create a custom pwm function
 */

/* NOTE: Includes */
#include "PulseWidthModulation.h"

#include <avr/io.h>

/* NOTE: Local declarations */
// TODO: None

/* NOTE: Global Variables */
// TODO: None

/* NOTE: Local function implementations */
void PWM_init(void)
{
    // set up PORTB.5 as an output and 0V
    DDRB |= 0x20;
    PORTB |= PORTB & ~0x20;

    // 512 @ 8
    // set frequency to 3900hz
    ICR1 = 512;

    // fast pwm set on compare
    TCCR1A = 0x02;
    // prescaler set to 8
    TCCR1B = 0x1a;
}

void PWM_dutyCycle(double percent)
{
    OCR1A = percent * 512;
}

void PWM_enable(void)
{
    TCCR1A |= 0x80;
}

void PWM_disable(void)
{
    OCR1A = 0x00;

    TCCR1A = (TCCR1A & ~0x80);
}

```

```c
/*
 * FileName: AnalogToDigital.h
 * Version: 1
 *
 * Created: 10/19/2022 12:47 AM
 * Author: Ethan Zeronik
 *
 * Operations: header for the adc submodule
 */

#ifndef AnalogToDigital_h_INCLUDED
#define AnalogToDigital_h_INCLUDED

#if defined(__cplusplus)
extern "C" {
#endif

#include "stdio.h"

#define F_CPU 16000000UL

/* NOTE: Custom Types */
// TODO: None

/* NOTE: Function prototypes */
// init registers for adc
void ADC_init(void);
// returns the value of the given channel
double ADC_getTenBitValue(uint16_t channel);

#if defined(__cplusplus)
} /* extern "C" */
#endif

#endif // AnalogToDigital_h_INCLUDED
```

```c
/*
 * FileName: AnalogToDigital.c
 * Version: 1
 *
 * Created: 10/19/2022 12:47 AM
 * Author: Ethan Zeronik
 *
 * Operations: basic adc implementation
 */

/* NOTE: Includes */
#include "AnalogToDigital.h"

#include <avr/io.h>

/* NOTE: Local declarations */
// TODO: None

/* NOTE: Global Variables */
// TODO: None

/* NOTE: Local function implementations */
void ADC_init(void)
{
    // ten bit one way mode
    ADCSRA = (1 << ADEN) | (1 << ADPS1) | (1 << ADPS0);

    // 5v reference
    ADMUX = (1 << REFS0);

    ADCSRB = 0x00;
}

double ADC_getTenBitValue(uint16_t channel)
{
    uint16_t result = 0;

    // select the channel
    ADMUX = (ADMUX & 0xe0) | channel;
    ADCSRB = (ADCSRB & 0xf7) | (channel >> 2);

    // start conversion
    ADCSRA |= (1 << ADSC);

    // wait for conversion
    while((ADCSRA & (1 << ADSC)) == 1)
    {
        // do nothing
    }

    // save result
    result = ADCL;
    result = result | (ADCH << 8);

    return result / 1024.0;
}
```