Waldo

3.1 Waldo Input

3.1.2

```
#include "MEAM_general.h"  // includes the resources included in the MEAM_general.h file
#include "m_usb.h" //  library used for printing

void setupADC(int ADCchnl){
  set(ADMUX,REFS0);//vcc
  set(ADCSRA,ADPS0);//1/128
  set(ADCSRA,ADPS1);
  set(ADCSRA,ADPS2);

  if(ADCchnl==0){
    set(DIDR0,ADC0D);//ADC0 disabling digital input
    clear (ADMUX,MUX0);// selecting single ended channel
    clear(ADMUX,MUX1);
    clear(ADMUX,MUX2);
    clear(ADCSRB,MUX5);
  }
  if(ADCchnl==1){
    set(DIDR0,ADC1D);//ADC1 disabling digital input
    // selecting single ended channel
    set (ADMUX,MUX0);
    clear(ADMUX,MUX1);
    clear(ADMUX,MUX2);
    clear(ADCSRB,MUX5);
  }
   if(ADCchnl==4){
    set(DIDR0,ADC4D);//ADC4 disabling digital input
    clear (ADMUX,MUX0);// selecting single ended channel
    clear(ADMUX,MUX1);
    set(ADMUX,MUX2);
    clear(ADCSRB,MUX5);
  }
   if(ADCchnl==5){
    set(DIDR0,ADC5D);//ADC5 disabling digital input
    set (ADMUX,MUX0);// selecting single ended channel
    clear(ADMUX,MUX1);
    set(ADMUX,MUX2);
    clear(ADCSRB,MUX5);
  }
```

```
 if(ADCchnl==6){
  set(DIDR0,ADC6D);//ADC6 disabling digital input
  clear (ADMUX,MUX0);// selecting single ended channel
  set(ADMUX,MUX1);
  set(ADMUX,MUX2);
  clear(ADCSRB,MUX5);
}

if(ADCchnl==7){
  set(DIDR0,ADC6D);//ADC7 disabling digital input
  set (ADMUX,MUX0);// selecting single ended channel
  set(ADMUX,MUX1);
  set(ADMUX,MUX2);
  clear(ADCSRB,MUX5);
}
if(ADCchnl==8){
  set(DIDR2,ADC8D);//ADC8 disabling digital input
  set(ADCSRB,MUX5);// selecting single ended channel
  clear (ADMUX,MUX0);
  clear(ADMUX,MUX1);
  clear(ADMUX,MUX2);
}
if(ADCchnl==9){
  set(DIDR2,ADC9D);//ADC9 disabling digital input
  set(ADCSRB,MUX5);// selecting single ended channel
  set (ADMUX,MUX0);
  clear(ADMUX,MUX1);
  clear(ADMUX,MUX2);
}
 if(ADCchnl==10){
  set(DIDR2,ADC10D);//ADC10 disabling digital input
  clear (ADMUX,MUX0);// selecting single ended channel
  set(ADMUX,MUX1);
  clear(ADMUX,MUX2);
  set(ADCSRB,MUX5);
}
```

```
  if(ADCchnl==11){
   set(DIDR2,ADC11D);//ADC11 disabling digital input
   set (ADMUX,MUX0);// selecting single ended channel
   set(ADMUX,MUX1);
   clear(ADMUX,MUX2);
   set(ADCSRB,MUX5);
  }
  if(ADCchnl==12){
   set(DIDR2,ADC12D);//ADC9 disabling digital input
   clear (ADMUX,MUX0);// selecting single ended channel
   clear(ADMUX,MUX1);
   set(ADMUX,MUX2);
   set(ADCSRB,MUX5);
  }

  if(ADCchnl==13){
   set(DIDR2,ADC13D);//ADC9 disabling digital input
   set (ADMUX,MUX0);// selecting single ended channel
   clear(ADMUX,MUX1);
   set(ADMUX,MUX2);
   set(ADCSRB,MUX5);

  }
  else{
   m_usb_tx_string( "Not an ADC channel");
  }
  set(ADCSRA,ADEN); // enabling the ADC subsystem
  set(ADCSRA,ADSC);// start conversion
  set(ADCSRA,ADIF); // reading the result
}
```

```
int ADCread(int ADCchnl){
   int tadc; // temporarily stores ADC value
   setupADC(ADCchnl);
   while(!bit_is_set(ADCSRA,ADIF)); // wait for bit to be set
   set(ADCSRA,ADIF);
   m_usb_tx_string( "potentiometer");m_usb_tx_uint(ADCchnl); m_usb_tx_string( "value = ");
   m_usb_tx_uint(ADC);
   m_usb_tx_string( "\n ");
   tadc=ADC;
   set(ADCSRA,ADSC);
   return tadc;

}

int main (void){
 m_usb_init();
 for (;;){
   ADCread(5);
 }
 return 0; // never reached
}
```

3.1.3

The door only moves through a 90 degree rotation while the gear on the lock needs a full 180 degrees to fully extend and retract the deadbolt.

The potentiometer of the door has a minimum raw value of 490 and a maximum raw value of 880.This means there are 390 ADC counts in a span of 90 degrees or 4.33 $ADC\ counts\ /degree$

.The potentiometer of the lock has a minimum raw value of 307 and a maximum raw value of 930.This means there are 627 ADC counts in a span of 180 degrees or 3.5 $ADC\ counts\ /degree$ .The ADC count seems to stay quite linear throughout the motion and there is some noise when the door gets stuck in the box.

The code for this was very similar to Q 3.1.1
The setupADC and ADCread subroutines are implemented without any changes.
We add two subroutines - convpot_lock(int) and convpot_door(int) to convert raw values to degrees.

```
void convpot_door(int raw){//pot9
  int max = 880; // taken from trials
  int min = 490;
  double t1;
  double t2 ;
  double degrees;
  t1 = abs(raw-min);
  t2 = (max-min);
  degrees = t1*90/t2; // converting raw values to degrees, door only moves 90 deg
  m_usb_tx_string( "door  = ");
  m_usb_tx_uint(degrees);
  m_usb_tx_string( " deg");
  m_usb_tx_string( "\n ");

}
 void convpot_lock(int raw){//pot5
  int max = 930;
  int min = 307;
  double t1;
  double t2 ;
  double degrees;
  t1 = abs(raw-min);
  t2 = (max-min);
  degrees = t1*180/t2; // converting raw values to degrees, lock gear moves 180 deg
  m_usb_tx_string( "lock = ");
  m_usb_tx_uint(degrees);
  m_usb_tx_string( " deg");
  m_usb_tx_string( "\n ");
}

int main (void){
  m_usb_init();
  for (;;){
    convpot_door(ADCread(5));
    convpot_lock(ADCread(9));
  }
  return 0; // never reached
}
```

[DEMO VIDEO]

3.1.1 Already submitted