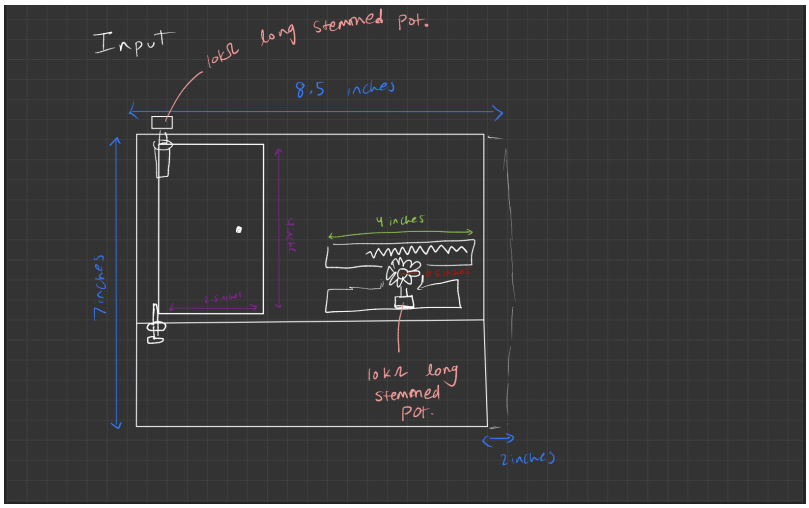
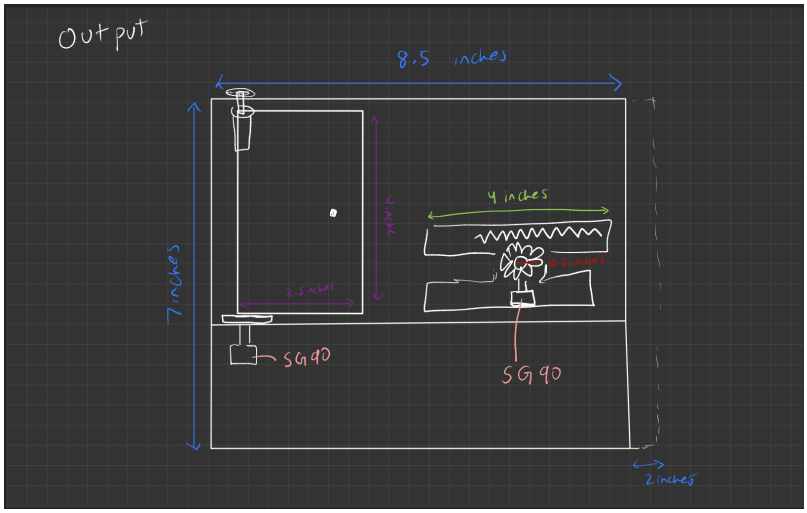
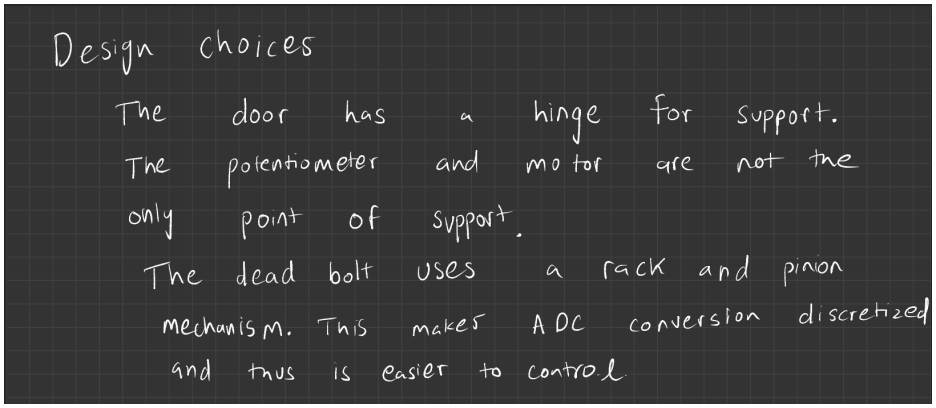
Waldo

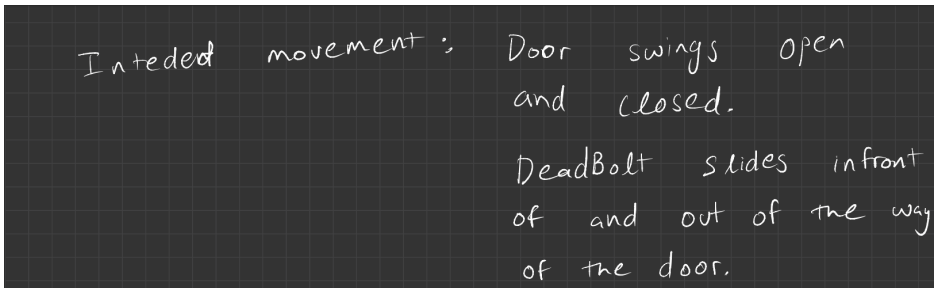
3.2 Waldo output

3.2.1







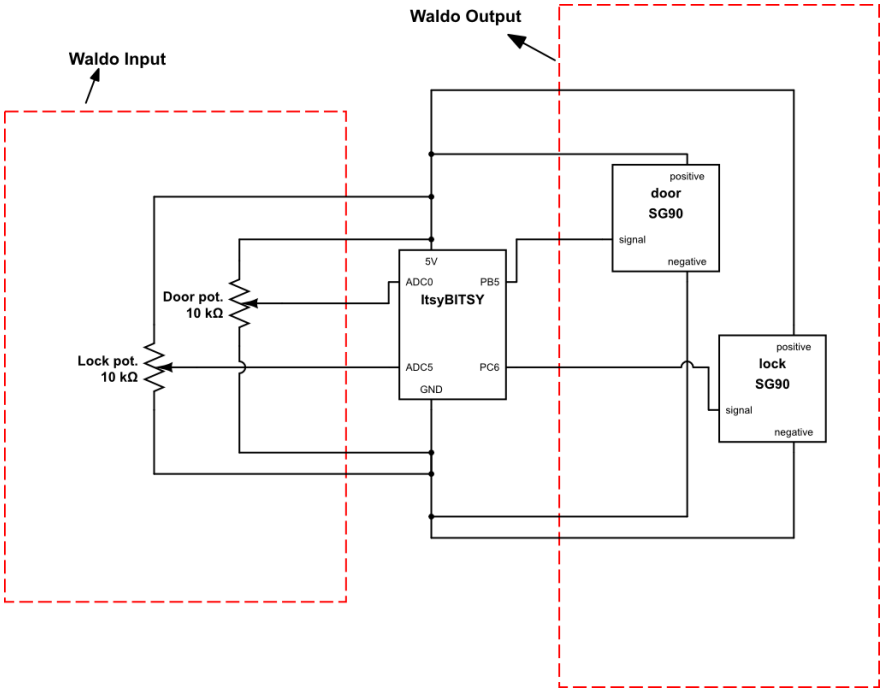


3.2.2

Worst case of current draw occurs at stall condition for servos. We can see that current draw at stall for SG90 is $360mA$ at $5V$ [1]. I am using 2 servos. The other component in the circuit are the potentiometers. I am using two $10k\Omega$ resistors and passing $5V$ across them. Hence the current drawn by each is $V/R = 5/10000 = 0.05mA$.
Therefore worst case scenario of total current drawn=> $(360 * 2) + (0.05 * 2) = 720.1\,mA$

This is not good as you can only draw $500mA$ from the Atmega - ItsyBitsy [2].

We will be okay as long as the stalling of the motors does not occur for long periods of time, because the typical current drawn during rotation by the servos is between $100mA - 250mA$, which would keep us within $500mA$ limit.


Waldo Output
Waldo Input
positive
door
SG90
signal
negative
5V
ADC0
PB5
ItsyBITSY
Door pot.
10 kΩ
Lock pot.
10 kΩ
ADC5
PC6
GND
positive
lock
SG90
signal
negative

3.2.3

Users > virgoyal > Desktop > lab > src > C 3.2S.c > main(void)

```c
/* Name: 3.1.2.c
 * Author: Vir Goyal
 */

#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
  clear(ADMUX,REFS1);
  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);
```

```
43      }
44       if(ADCchnl==6){
45        set(DIDR0,ADC6D);//ADC6 disabling digital input
46        clear (ADMUX,MUX0);// selecting single ended channel
47        set(ADMUX,MUX1);
48        set(ADMUX,MUX2);
49        clear(ADCSRB,MUX5);
50      }
51
52      if(ADCchnl==7){
53        set(DIDR0,ADC6D);//ADC7 disabling digital input
54        set (ADMUX,MUX0);// selecting single ended channel
55        set(ADMUX,MUX1);
56        set(ADMUX,MUX2);
57        clear(ADCSRB,MUX5);
58      }
59      if(ADCchnl==8){
60        set(DIDR2,ADC8D);//ADC8 disabling digital input
61        set(ADCSRB,MUX5);// selecting single ended channel
62        clear (ADMUX,MUX0);
63        clear(ADMUX,MUX1);
64        clear(ADMUX,MUX2);
65      }
66      if(ADCchnl==9){
67        set(DIDR2,ADC9D);//ADC9 disabling digital input
68        set(ADCSRB,MUX5);// selecting single ended channel
69        set (ADMUX,MUX0);
70        clear(ADMUX,MUX1);
71        clear(ADMUX,MUX2);
72      }
73       if(ADCchnl==10){
74        set(DIDR2,ADC10D);//ADC10 disabling digital input
75        clear (ADMUX,MUX0);// selecting single ended channel
76        set(ADMUX,MUX1);
77        clear(ADMUX,MUX2);
78        set(ADCSRB,MUX5);
79      }
80       if(ADCchnl==11){
81        set(DIDR2,ADC11D);//ADC11 disabling digital input
82        set (ADMUX,MUX0);// selecting single ended channel
83        set(ADMUX,MUX1);
84        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);

  }

}
  int ADCreadv(){

    int tadc; // temporarily stores ADC value
    set(ADCSRA,ADEN); // enabling the ADC subsystem
    set(ADCSRA,ADSC);// start conversion

    while(!bit_is_set(ADCSRA,ADIF)); // wait for bit to be set
    tadc=ADC;
    return tadc;
    set(ADCSRA,ADIF); // clear the conversion flag
    set(ADCSRA,ADSC); //start converting again

  }

```

```
121  int main (void){
122
123  m_usb_init();
124
125
126  int door;
127  int lock;
128    for (;;){
129      setupADC(); // initialise ADC
130      door=ADCreadv();
131      m_usb_tx_string( "door = ");
132      m_usb_tx_uint(door);
133      m_usb_tx_string( "\t ");
134      setupADC(5);
135      lock =ADCreadv();  // initialise ADC
136      m_usb_tx_string( "lock = ");
137      m_usb_tx_uint(lock);
138      m_usb_tx_string( "\n ");
139
140      set(DDRC,6);
141      set(PORTC,6);
142
143      set(DDRB,5);
144      set(PORTB,5);
145
146      _clockdivide(0);
147
148    // set timer on OCR3A mode 7
149      set(TCCR3B,WGM32);
150      set(TCCR3A,WGM30);
151      clear(TCCR3B,WGM33);
152      set(TCCR3A,WGM31);
153
154      set(TCCR3B, CS30);
155      clear(TCCR3B, CS31);
156      set(TCCR3B, CS32); // set prescaler to /1024
157
158
159    //clear at OCR3A, set at rollover
160    set (TCCR3A,COM3A1);
161    clear (TCCR3A,COM3A0);
```

```
164     // set timer on OCR1B mode 7
165     set(TCCR1B,WGM12);
166     set(TCCR1A,WGM10);
167     clear(TCCR1B,WGM13);
168     set(TCCR1A,WGM11);
169
170     set (TCCR1B,CS10); //Setting prescaler to 1/1024
171     set (TCCR1B,CS12);
172     clear (TCCR1B,CS11);
173
174     //clear at OCR1A, set at rollover
175     set (TCCR1A,COM1A1);
176     clear (TCCR1A,COM1A0);
177
178     float compareval_door = 0.055*door;
179     OCR1A = compareval_door;
180
181     float compareval_lock = 0.055*lock;
182     OCR3A = compareval_lock;
183
184     }
185     return 0; // never reached
186   }
```

DEMO VIDEO

3.2.4

DANCE