CSE211s:Introduction to Embedded Systems

Lect. #8: P-controller for single tank system

Ahmed M. Zaki

Mathematical Mode of a single tank

$$\frac{H(s)}{Q_i(s)} = \frac{R}{RCs+1}$$

If R=1

$$\frac{H(s)}{Q_i(s)} = \frac{1}{Cs+1}$$

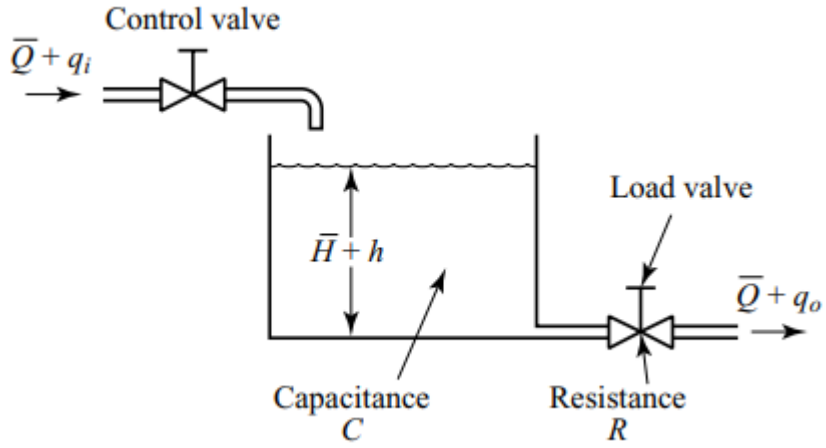
$$\frac{E_o(s)}{E_i(s)} = \frac{1}{RCs+1}$$

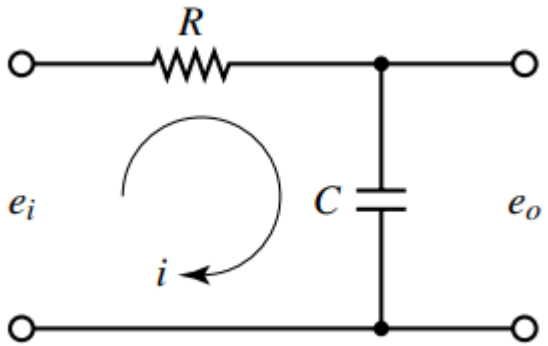
If R=1

$$\frac{E_o(s)}{E_i(s)} = \frac{1}{Cs+1}$$

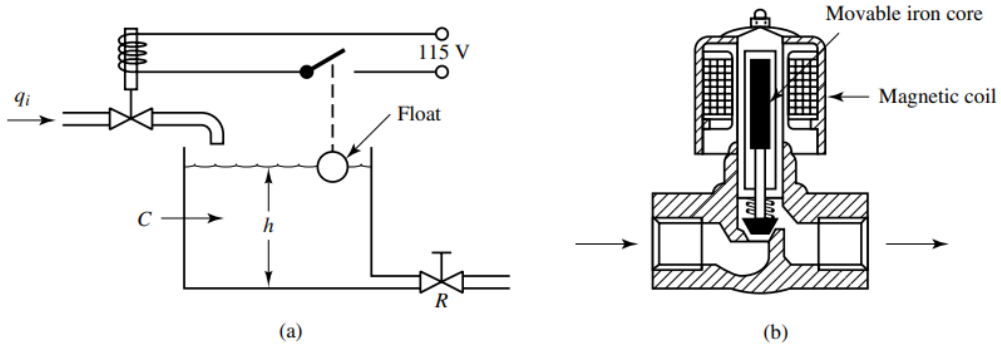
We can consider

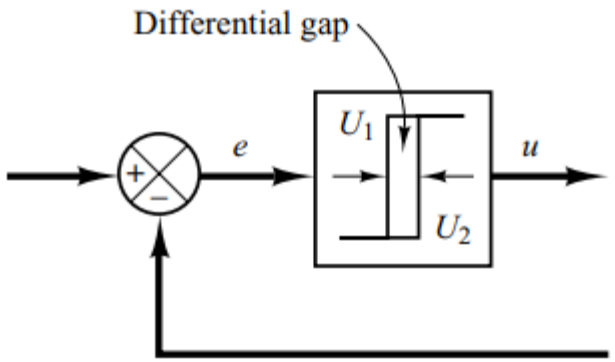
$E_o(s)\ act\ as\ \ H(s)$, and $E_i(s)\ act\ as$ Q(s) : voltage to the pump

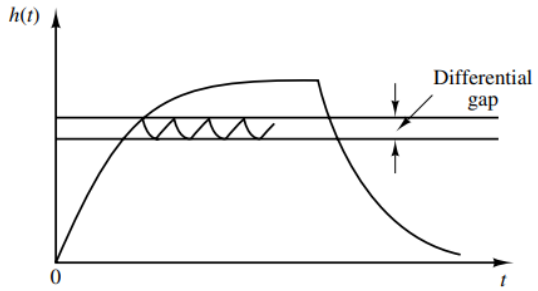




On-off controller for single tank







Proportional Control Action

Proportional Control Action. For a controller with proportional control action, the relationship between the output of the controller $u(t)$ and the actuating error signal $e(t)$ is

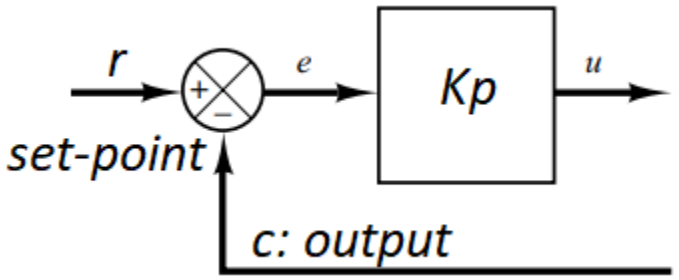
$$u(t) = K_p e(t)$$

or, in Laplace-transformed quantities,

$$\frac{U(s)}{E(s)} = K_p$$

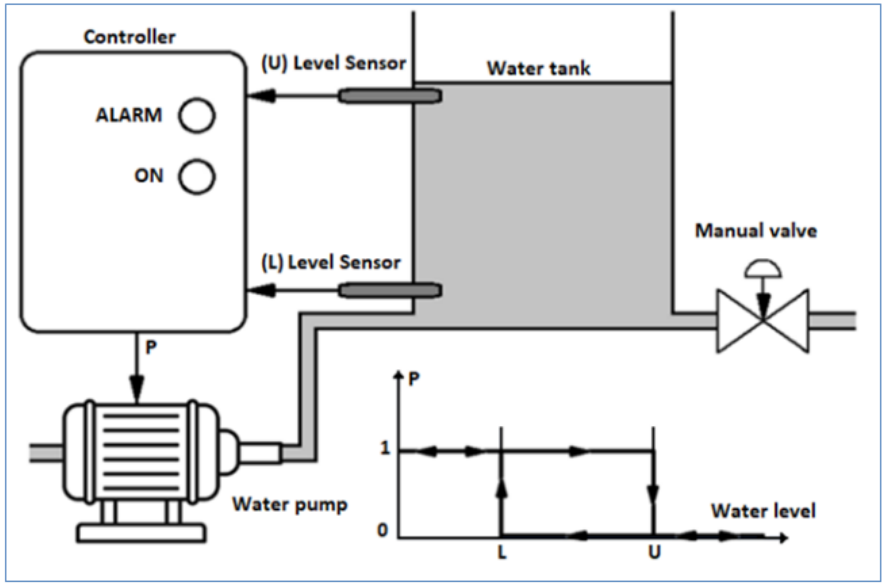
where K_p is termed the proportional gain.

Whatever the actual mechanism may be and whatever the form of the operating power, the proportional controller is essentially an amplifier with an adjustable gain.

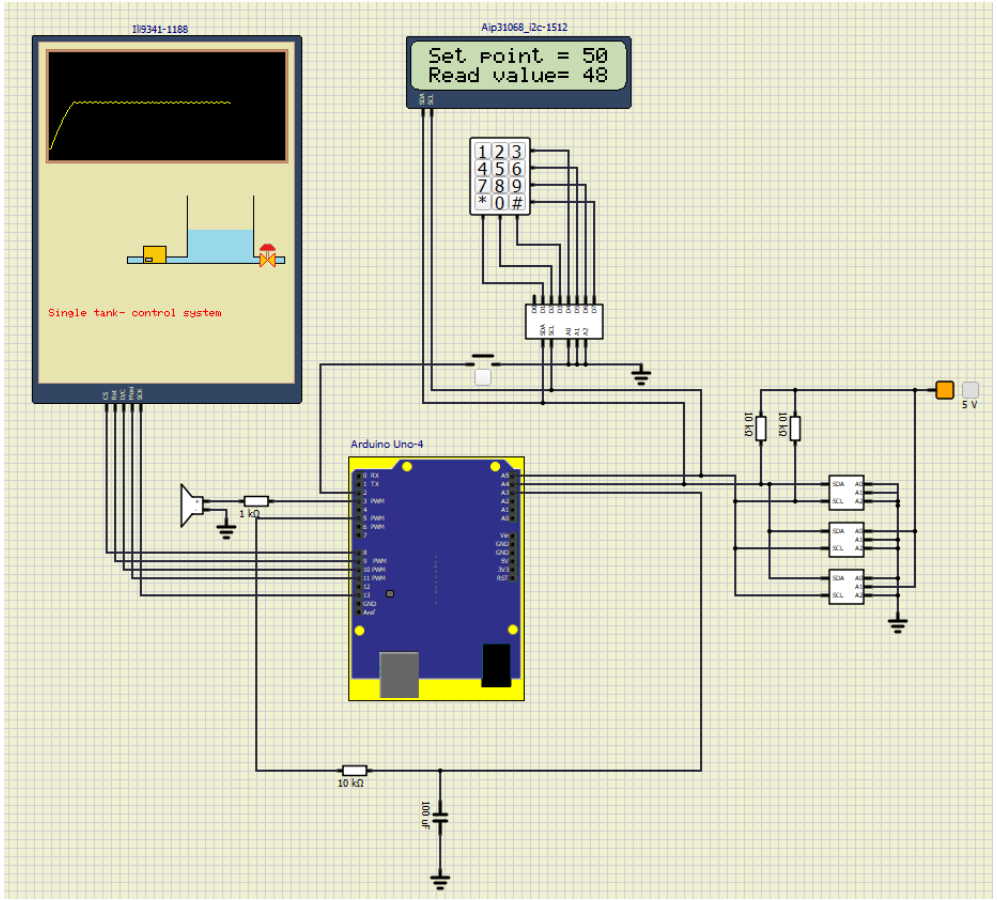


Exam 2020/2021

Q14—Q24: The figure to the right shows the connection between a water tank and its level Controller. The controller has two inputs: Upper-Level Sensor (U) and Lower-Level Sensor (L), which sense the level of the water inside the tank. Both sensors are connected through signal conditioning circuits to output logic high/low voltage levels. Controller output (P) is used to control a water pump. ON (Green) and ALARM (Red) LED indicators are used to show the state of the pump. The controller runs PROG1 (on the next page) on a TM4C microcontroller. The pump should be turned on (by applying a logic high on signal P), if the water level in the tank is below the low level until reaching the upper level, then it is turned off until the water level drops again below the lower level. If it happens that the pump is on for a specific time duration without having the water reaching the upper level, the pump is turned off and the ALARM LED is switched on for a specific duration. Then, the pump is switched on again (with ALARM LED off), continuing as normal.

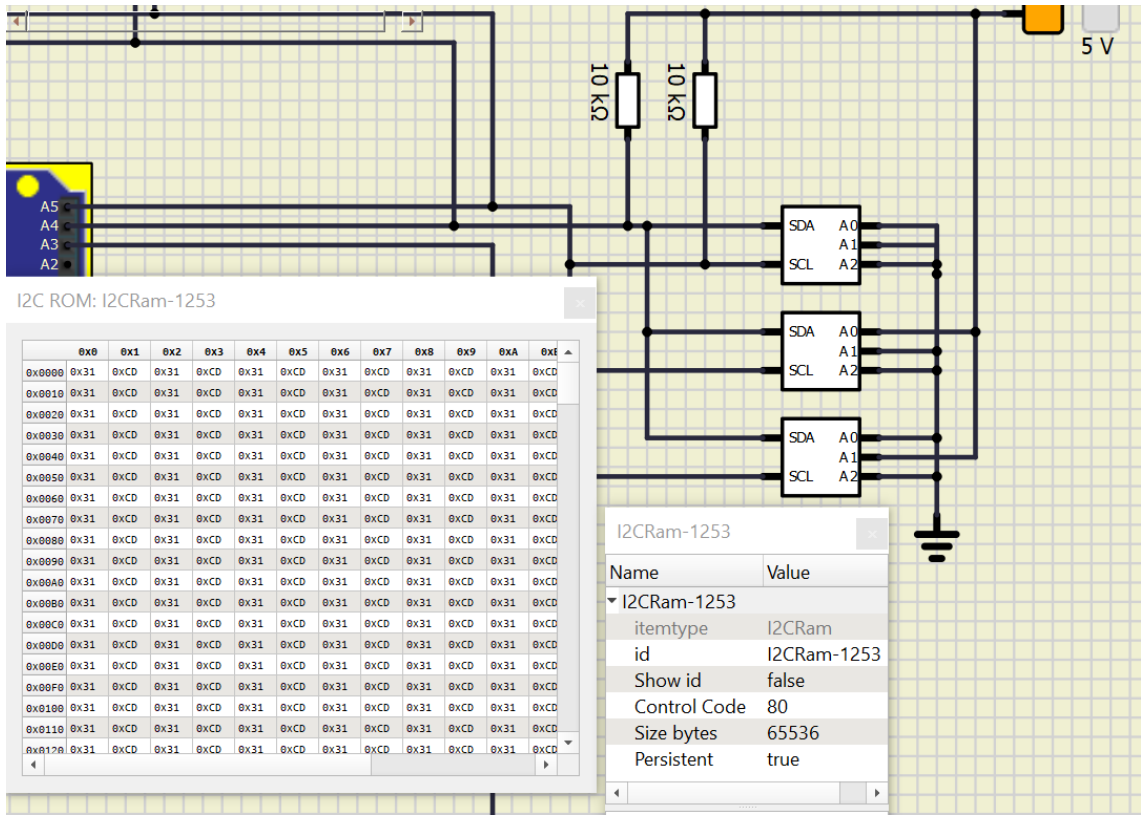


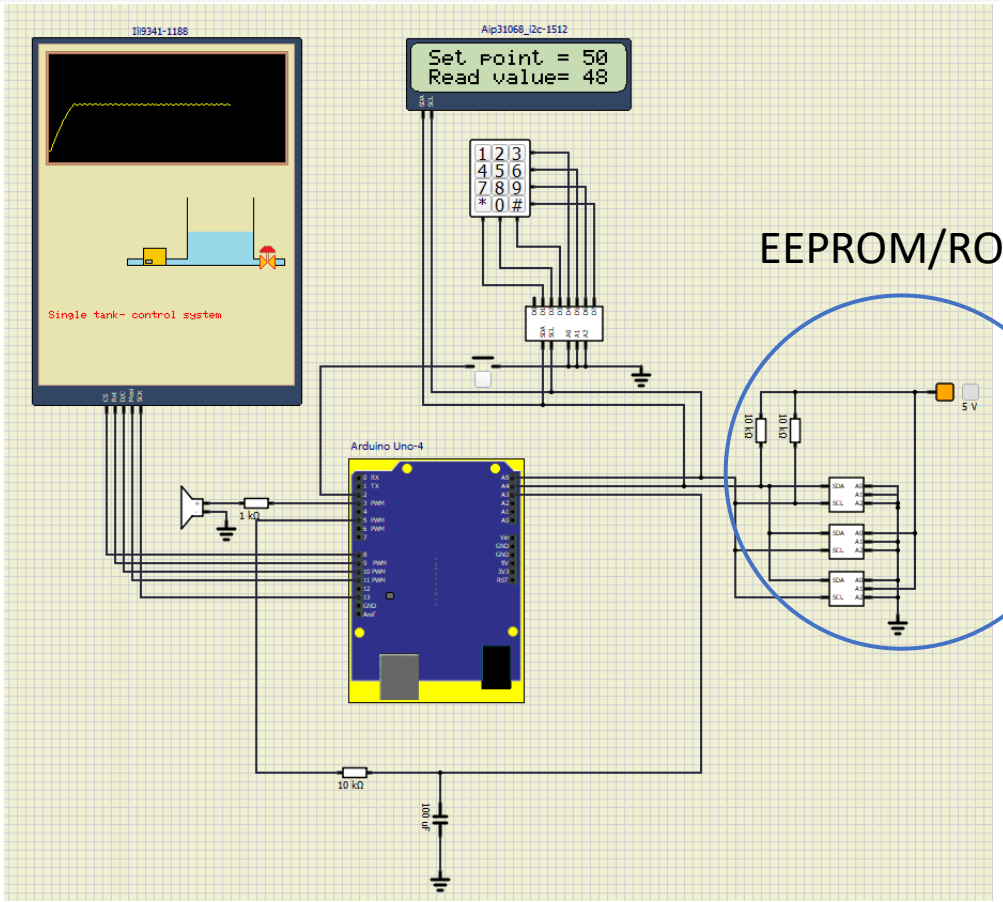
P-Controller for Single Tank system



P-Controller for Single Tank system

EEPROM/ROM

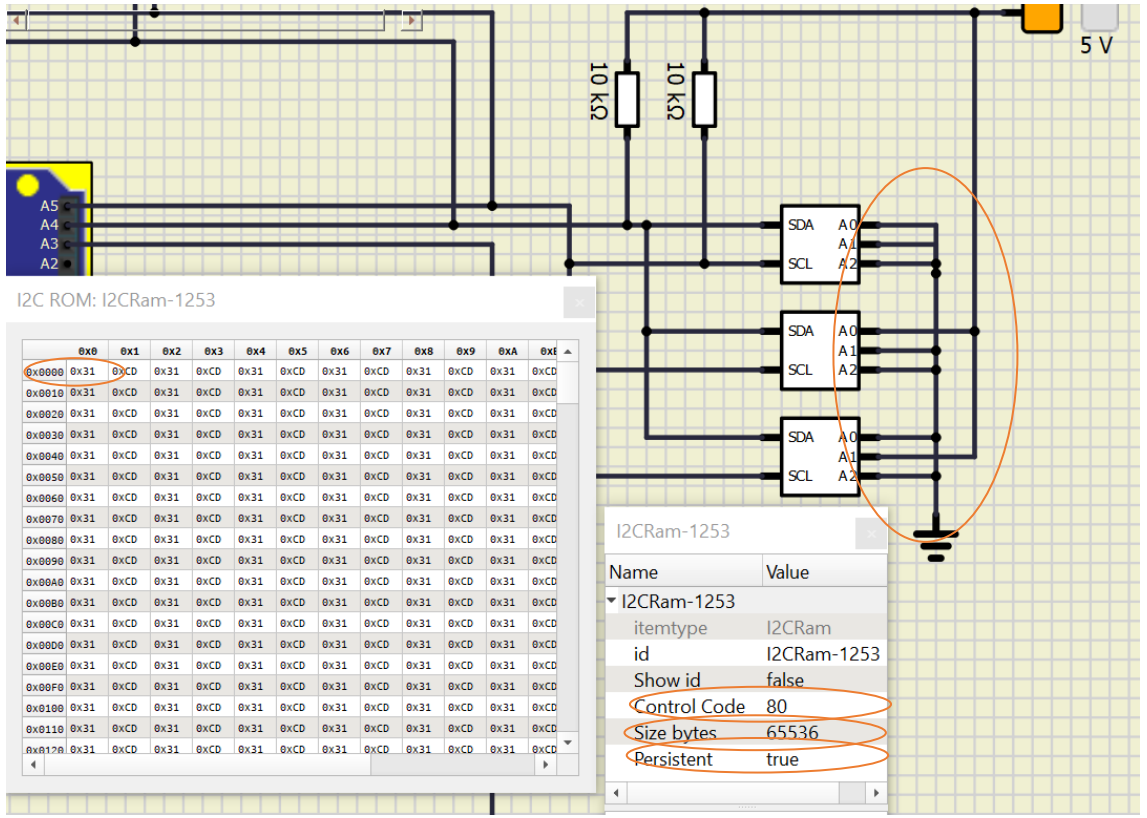




P-Controller for Single Tank system

EEPROM/ROM Code

```
#define ROM_ADDRESS 80
#define NUM_OF_WORD 16
#define WORD_SIZE 2
void MEMRead(unsigned long address ,unsigned int *data, unsigned int len ){
    int read_val1;
    int read_val2;
    int MSB_address;
    MSB_address=(address & 0xF0000)>>16;
    int MEM_ID=ROM_ADDRESS+MSB_address;
    Wire.beginTransmission(MEM_ID);
    Wire.write((address & 0xFF00)>>8);  //MSB
    Wire.write(address & 0x00FF);       //LSB
    Wire.endTransmission();
    Wire.requestFrom(MEM_ID,len*WORD_SIZE);
    for  (int i=0;i<len;i++){
        read_val1=Wire.read();
        read_val2=Wire.read();
        data[i]= (read_val1)+(read_val2<<8);
    }
}
void setup() {
   paint_background();
}
unsigned int data[NUM_OF_WORD];
void paint_background(void){
    unsigned long address=0;
    int word_index=0;
    for  (int row=0;row<IMAGE_W;row++)  {
        for (int k=0;k<IMAGE_H/NUM_OF_WORD;k++) {
            MEMRead(address,data,NUM_OF_WORD);
            word_index=0;
            for  (int col=0;col<NUM_OF_WORD;col++){
                tft.drawPixel(row,col+(k*NUM_OF_WORD),data[word_index++]);
            }
            address+=2*NUM_OF_WORD;
        }
    }
```



P-Controller for Single Tank system

EEPROM/ROM Code (incorrect code/Fix)

```
#define ROM_ADDRESS 82
#define NUM_OF_WORD 16
#define WORD_SIZE 2
void MEMRead(unsigned long address ,unsigned int data, unsigned int len ){
    int read_val1;
    int read_val2;
    int MSB_address;
    MSB_address=(address & 0xF00)<<8;
    int MEM_ID=ROM_ADDRESS;
    Wire.beginTransmission(address);
    Wire.write((MEM_ID & 0xFF)<<8);     //MSB
    Wire.write(MEM_ID & 0x00FF);        //LSB
    Wire.endTransmission();
    Wire.requestFrom(MEM_ID,len);
    for  (int i=0;i<len;i++){
        read_val1=Wire.read();
        read_val2=Wire.read();
        data[i]= (read_val1)+(read_val2<<0xFF);
    }
}
void setup() {
   paint_background();
}
unsigned char data[NUM_OF_WORD];
void paint_background(void){
    unsigned long address=0;
    int word_index=0;
    for  (int row=0;row<IMAGE_W;row++)  {
        for (int k=0;k<IMAGE_H/NUM_OF_WORD;k++) {
            MEMRead(address,data[NUM_OF_WORD],NUM_OF_WORD);
            word_index=0;
            for  (int col=0;col<NUM_OF_WORD;col++){
                tft.drawPixel(row,col+(k*NUM_OF_WORD),data[word_index++]);
            }
            address+=2*NUM_OF_WORD;
        }
    }
```

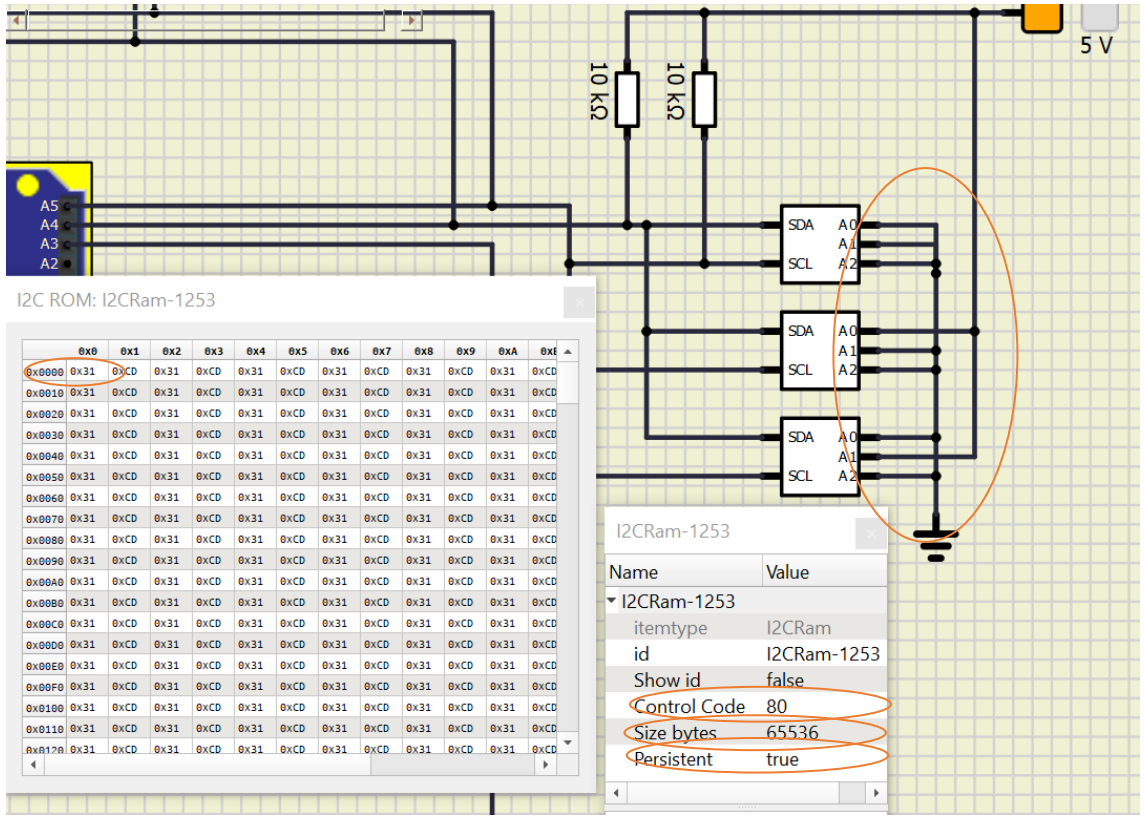
```
#define ROM_ADDRESS 80
#define NUM_OF_WORD 16
#define WORD_SIZE 2
void MEMRead(unsigned long address ,unsigned int *data, unsigned int len ){
    int read_val1;
    int read_val2;
    int MSB_address;
    MSB_address=(address & 0xF0000)>>16;
    int MEM_ID=ROM_ADDRESS+MSB_address;
    Wire.beginTransmission(MEM_ID);
    Wire.write((address & 0xFF00)>>8);  //MSB
    Wire.write(address & 0x00FF);       //LSB
    Wire.endTransmission();
    Wire.requestFrom(MEM_ID,len*WORD_SIZE);
    for  (int i=0;i<len;i++){
        read_val1=Wire.read();
        read_val2=Wire.read();
        data[i]= (read_val1)+(read_val2<<8);
    }
}
void setup() {
   paint_background();
}
unsigned int data[NUM_OF_WORD];
void paint_background(void){
    unsigned long address=0;
    int word_index=0;
    for  (int row=0;row<IMAGE_W;row++)  {
        for (int k=0;k<IMAGE_H/NUM_OF_WORD;k++) {
            MEMRead(address,data,NUM_OF_WORD);
            word_index=0;
            for  (int col=0;col<NUM_OF_WORD;col++){
                tft.drawPixel(row,col+(k*NUM_OF_WORD),data[word_index++]);
            }
            address+=2*NUM_OF_WORD;
        }
    }
```

P-Controller for Single Tank system

EEPROM/ROM Code (memory content) (RGB 888 to RGB 565)

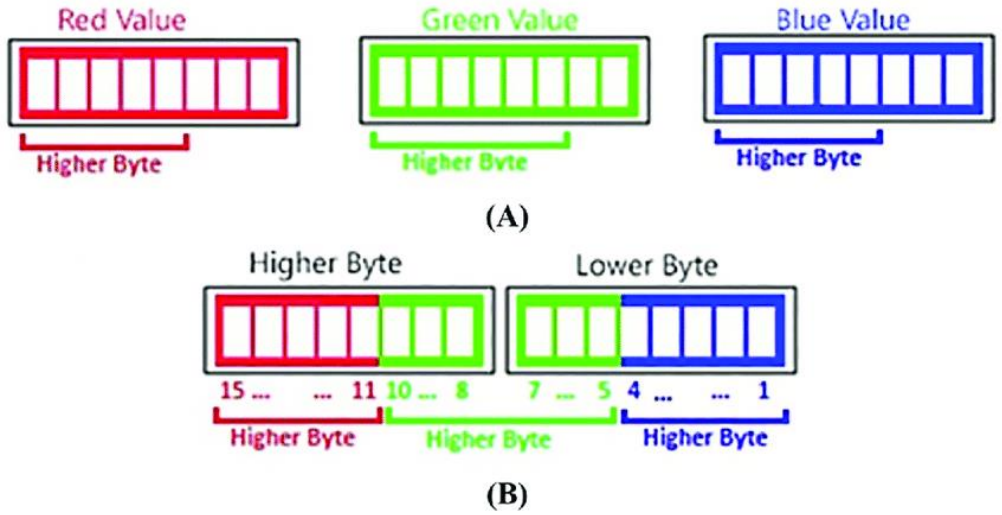
RGB 888

1		RGB(0,0,0)	29		RGB(128,0,128)
2		RGB(255,255,255)	30		RGB(128,0,0)
3		RGB(255,0,0)	31		RGB(0,128,128)
4		RGB(0,255,0)	32		RGB(0,0,255)
5		RGB(0,0,255)	33		RGB(0,204,255)
6		RGB(255,255,0)	34		RGB(204,255,255)
7		RGB(255,0,255)	35		RGB(204,255,204)
8		RGB(0,255,255)	36		RGB(255,255,153)
9		RGB(128,0,0)	37		RGB(153,204,255)
10		RGB(0,128,0)	38		RGB(255,153,204)
11		RGB(0,0,128)	39		RGB(204,153,255)
12		RGB(128,128,0)	40		RGB(255,204,153)
13		RGB(128,0,128)	41		RGB(51,102,255)
14		RGB(0,128,128)	42		RGB(51,204,204)
15		RGB(192,192,192)	43		RGB(153,204,0)
16		RGB(128,128,128)	44		RGB(255,204,0)
17		RGB(153,153,255)	45		RGB(255,153,0)
18		RGB(153,51,102)	46		RGB(255,102,0)
19		RGB(255,255,204)	47		RGB(102,102,153)
20		RGB(204,255,255)	48		RGB(150,150,150)
21		RGB(102,0,102)	49		RGB(0,51,102)
22		RGB(255,128,128)	50		RGB(51,153,102)
23		RGB(0,102,204)	51		RGB(0,51,0)
24		RGB(204,204,255)	52		RGB(51,51,0)
25		RGB(0,0,128)	53		RGB(153,51,0)
26		RGB(255,0,255)	54		RGB(153,51,102)
27		RGB(255,255,0)	55		RGB(51,51,153)
28		RGB(0,255,255)	56		RGB(51,51,51)



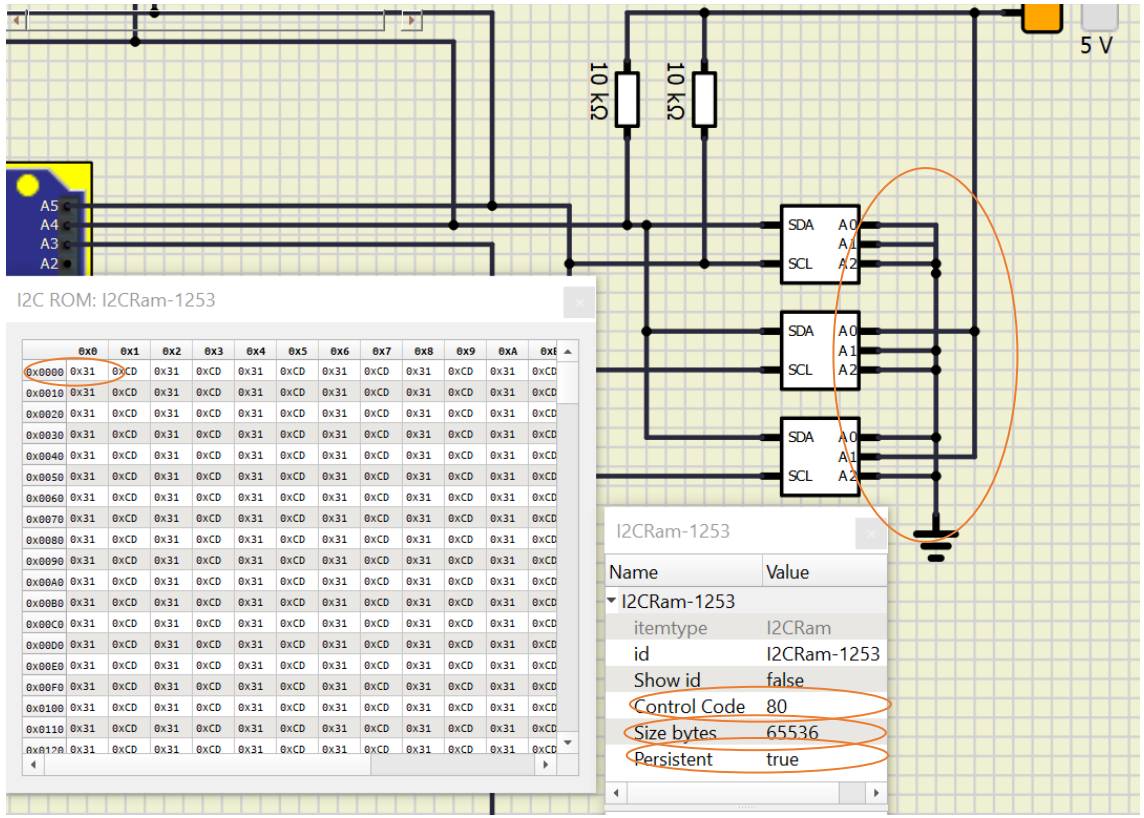
P-Controller for Single Tank system

EEPROM/ROM Code (memory content) (RGB 888 to RGB 565)



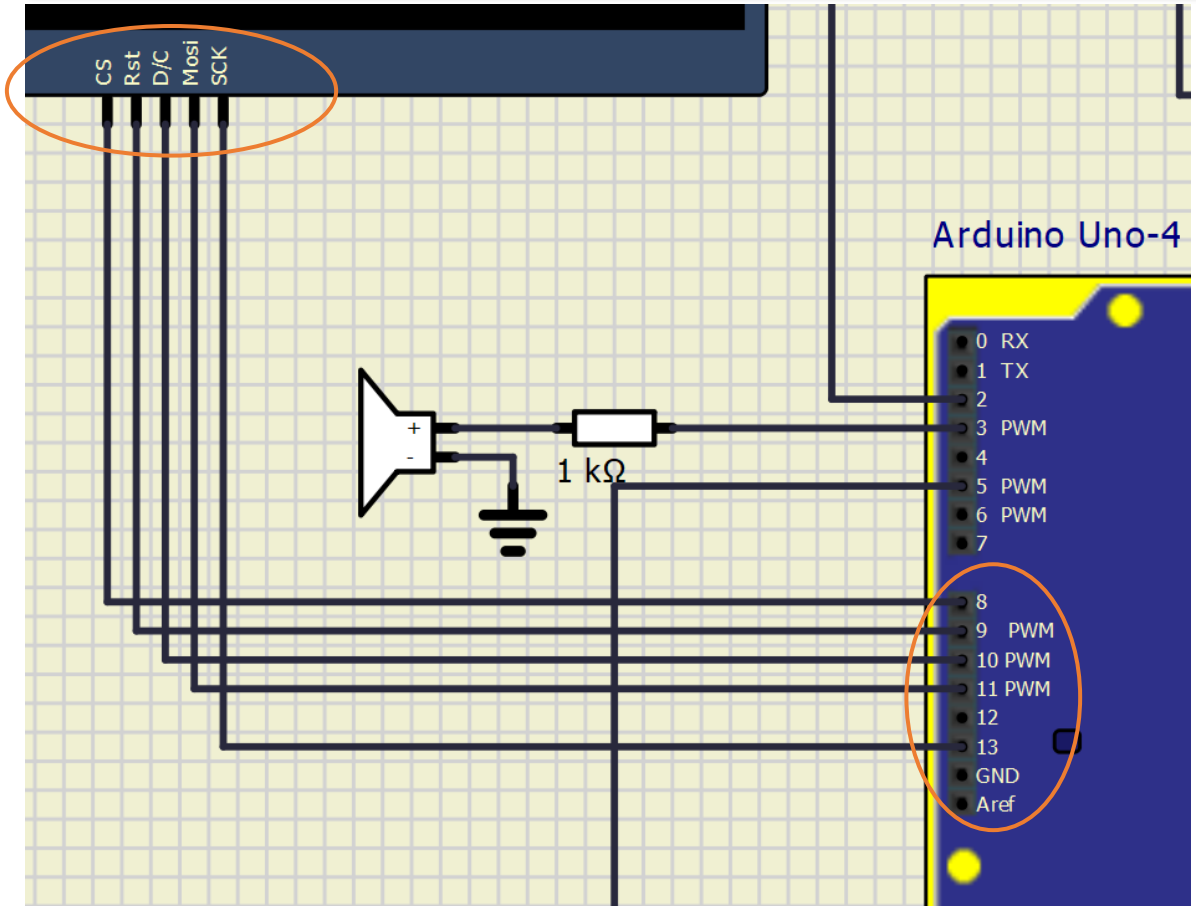
MATLAB code

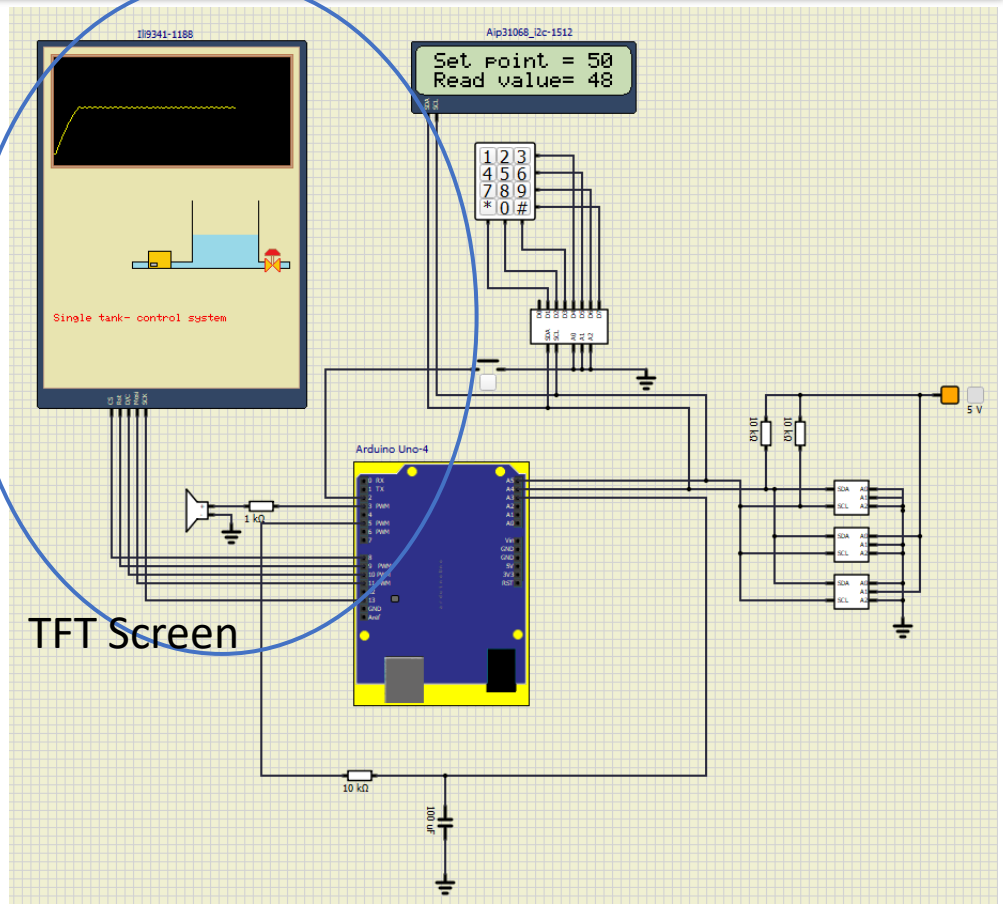
```
function ret=RGB888to565(R,G,B)
B= bitshift(B,-3);
G= bitshift(bitshift(G,-2),5);
R= bitshift(bitshift(R,-3),11);
ret=R+G+B;
```



P-Controller for Single Tank system

TFT Screen/SPI

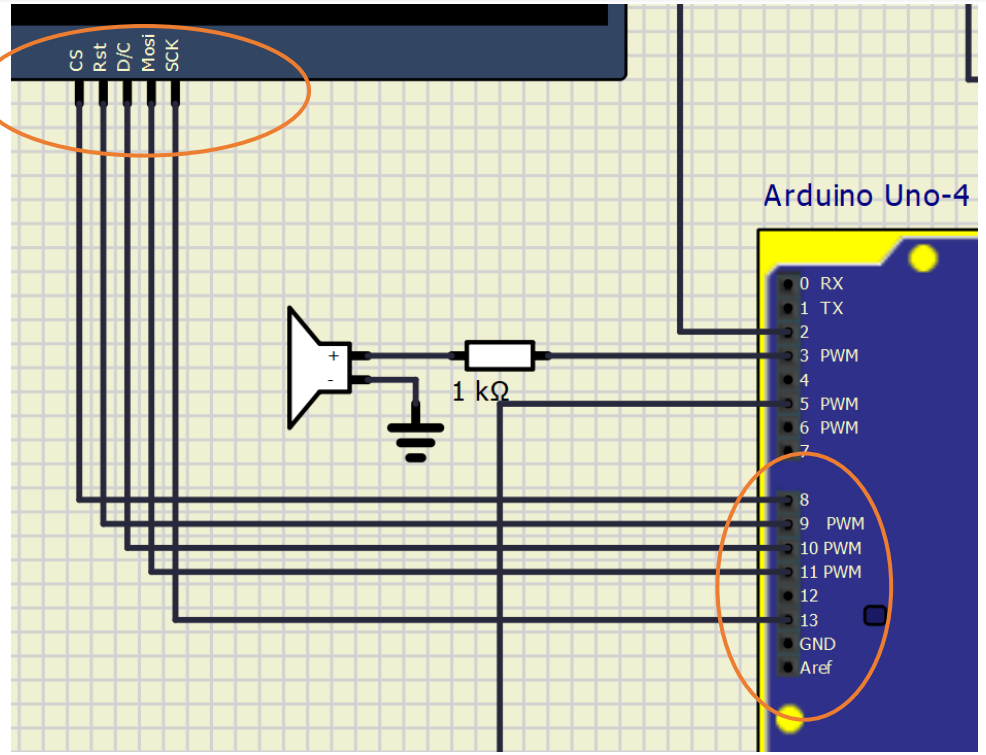




P-Controller for Single Tank system

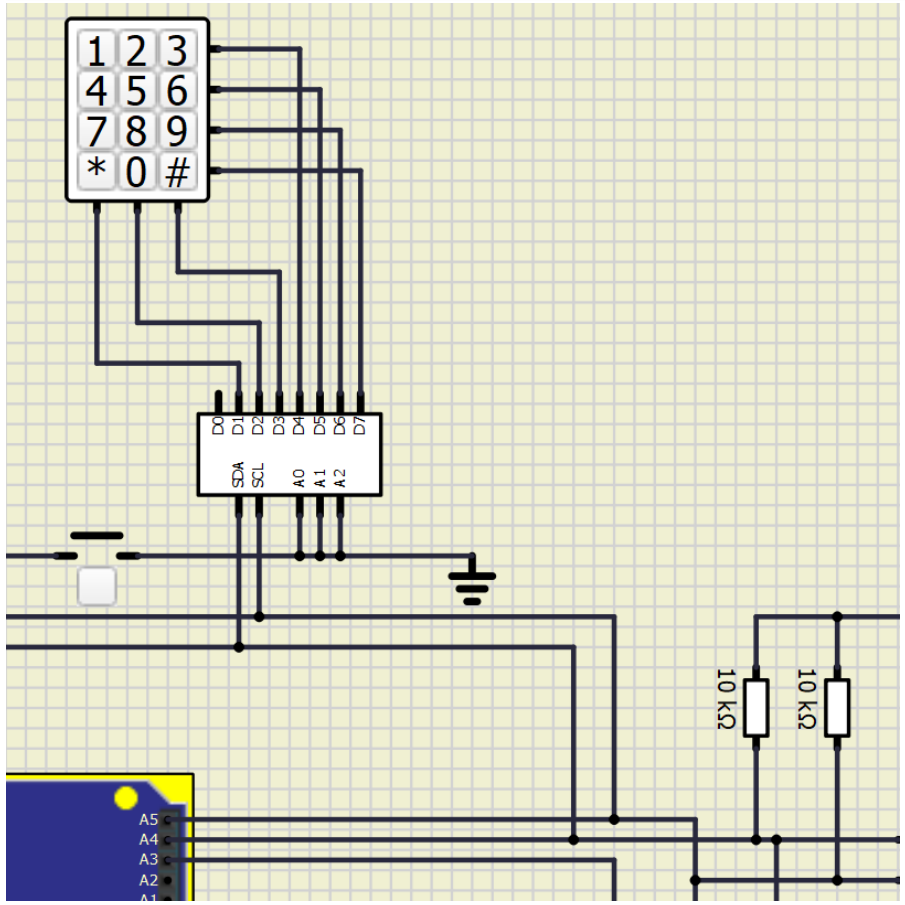
TFT Screen/SPI

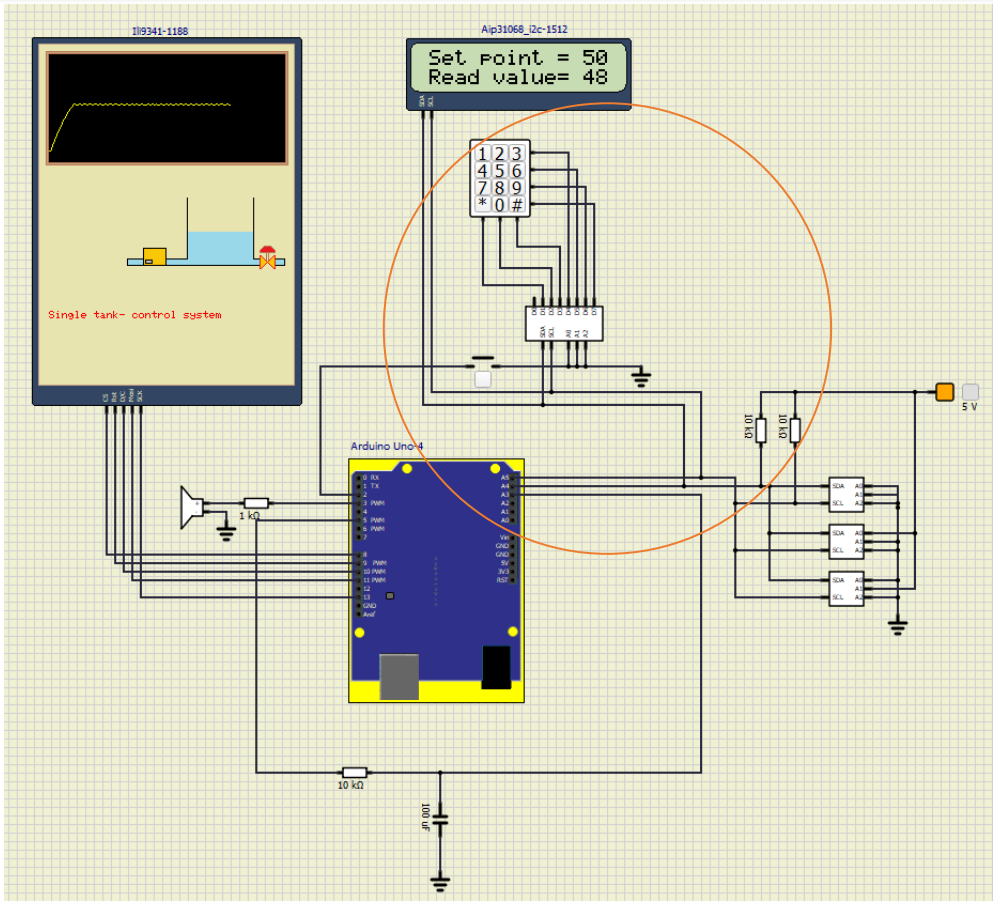
```
#include <Adafruit_ILI9341.h>
#define TFT_CS   8
#define TFT_RST  9
#define TFT_DC   10
#define TFT_MOSI 11
#define TFT_MISO 12  // not used
#define TFT_CLK  13

Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC,TFT_MOSI,TFT_CLK,TFT_RST,TFT_MISO);
void paint_back_ground(void){
    unsigned long address=0;
    int word_index=0;
    for  (int row=0;row<IMAGE_W;row++)  {
        for (int k=0;k<IMAGE_H/NUM_OF_WORD;k++) {
            MEMRead(address,data,NUM_OF_WORD);
            word_index=0;
            for  (int col=0;col<NUM_OF_WORD;col++){
                tft.drawPixel(row,col+(k*NUM_OF_WORD),data[word_index++]);
            }
            address+=2*NUM_OF_WORD;
        }
    }
}
byte current_water_level=0;
#define WATER_COLOR  0x9EDD
#define BACKGROUND_COLOR 0xEF36
void set_water_level(byte new_level_i)
{
    int new_level=new_level_i/2;
    for (int i=current_water_level;i>new_level;i--)
        tft.drawLine(140,200-i,200,200-i,BACKGROUND_COLOR);
    for (int i=current_water_level;i<=new_level;i++)
        tft.drawLine(140,200-i,200,200-i,WATER_COLOR);
    current_water_level=new_level;
}
```



P-Controller for Single Tank system

Keypad





P-Controller for Single Tank system

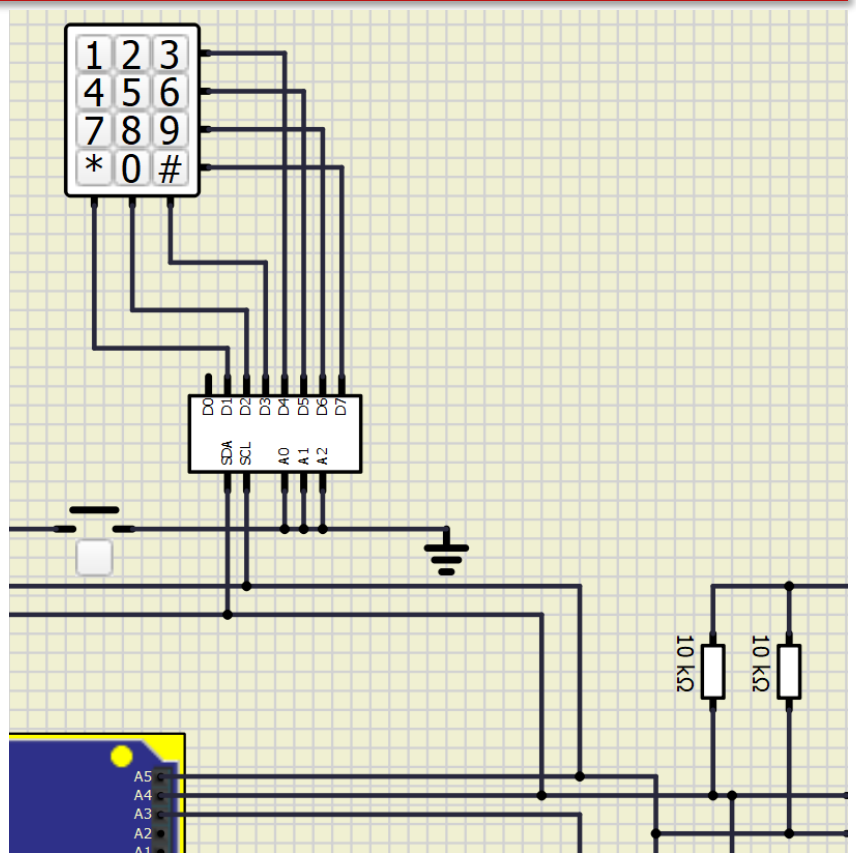
Keypad

```
#include <Wire.h>
byte get_key(void){
    byte MSB_Key_code[3]={0xfd,0xfb,0xf7};
    byte key=100; // nothing any number outside the keys
    byte LSB_Key_code;
    while(key==100){
        for (int k=0;k<3;k++){
            Wire.beginTransmission(85); Wire.write(MSB_Key_code[k]); Wire.endTransmission(); Wire.requestFrom(85,1);
            LSB_Key_code=Wire.read();
            LSB_Key_code=(((~(LSB_Key_code&0xf0))>>4)&0xf);
            if(k==0){
                if (LSB_Key_code==1)       {    key=1;
                } else if (LSB_Key_code==2){    key=4;
                } else if (LSB_Key_code==4){    key=7;
                } else if (LSB_Key_code==8){    key=10; //"*"
                }
            }else if(k==1){
                if (LSB_Key_code==1)       {    key=2;
                } else if (LSB_Key_code==2){    key=5;
                } else if (LSB_Key_code==4){    key=8;
                } else if (LSB_Key_code==8){    key=0;
                }
            }else if(k==2){
                if (LSB_Key_code==1)       {    key=3;
                } else if (LSB_Key_code==2){    key=6;
                } else if (LSB_Key_code==4){    key=9;
                } else if (LSB_Key_code==8){    key=20; //"#"
                }
            }
        }
    }
    LSB_Key_code=0;
    tone(AUDIO_OUT,100*key,200);delay(200);
    while(LSB_Key_code!=0xF0) {
            Wire.beginTransmission(85); Wire.write(0xf0); Wire.endTransmission(); Wire.requestFrom(85,1);
            LSB_Key_code=Wire.read();
    }
    return key;
}
```

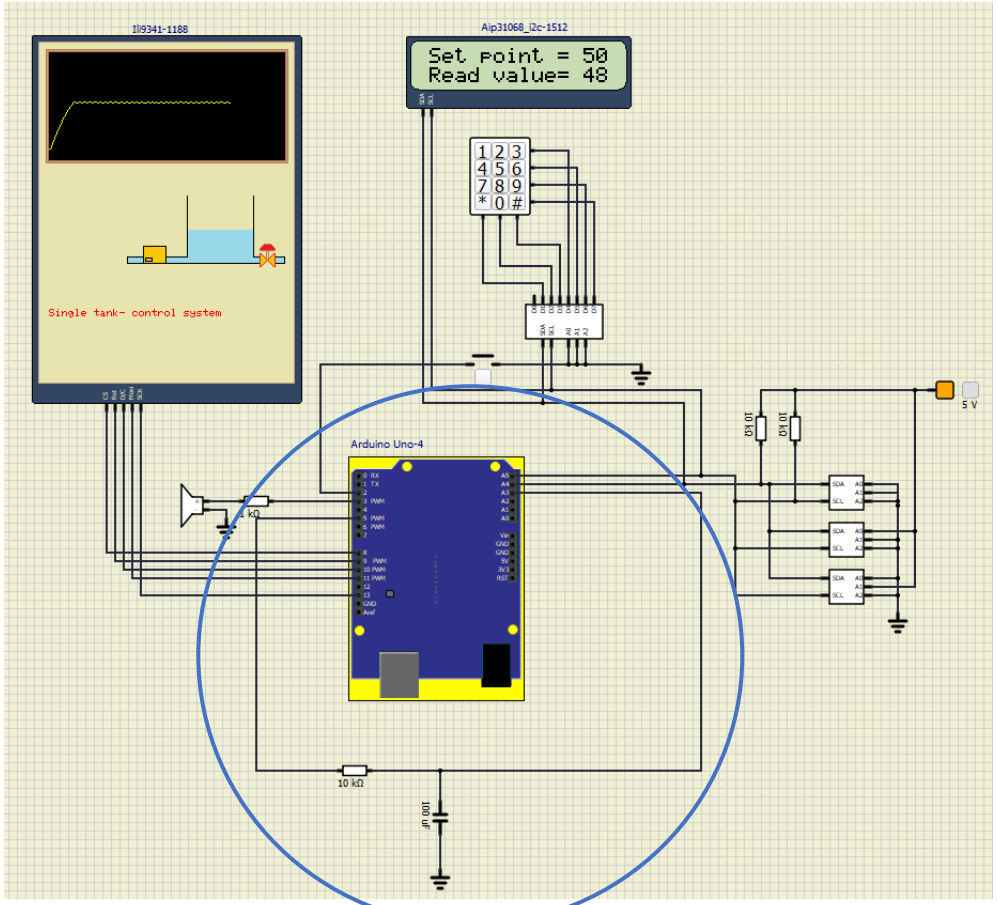
0xfd = 1111 - 1101

0xfb = 1111 - 1011

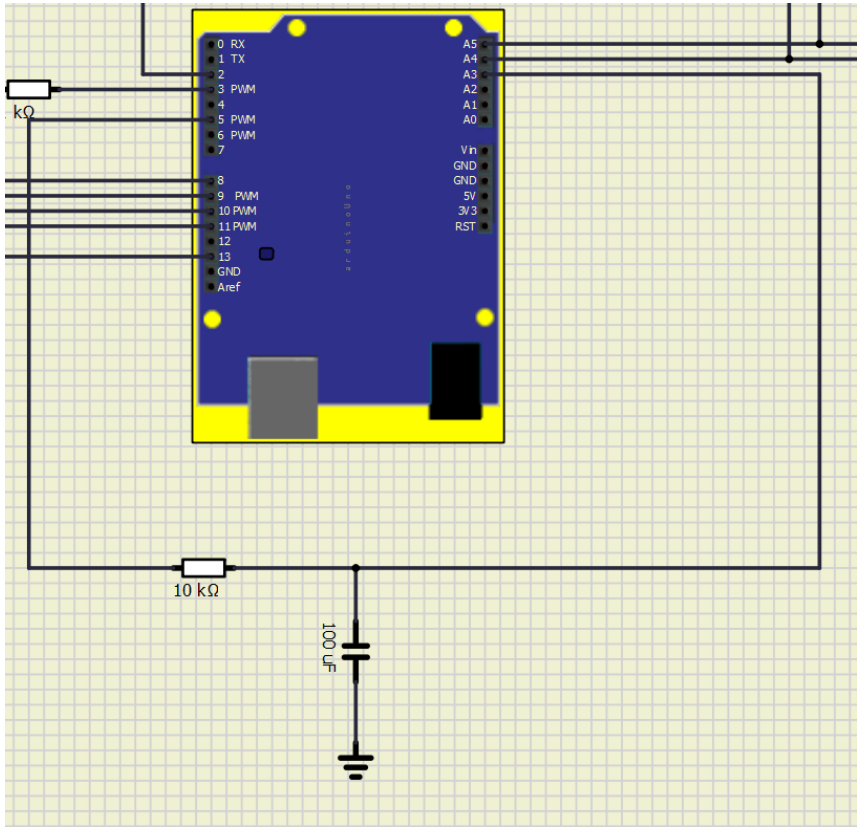
0xf7 = 1111 - 0111

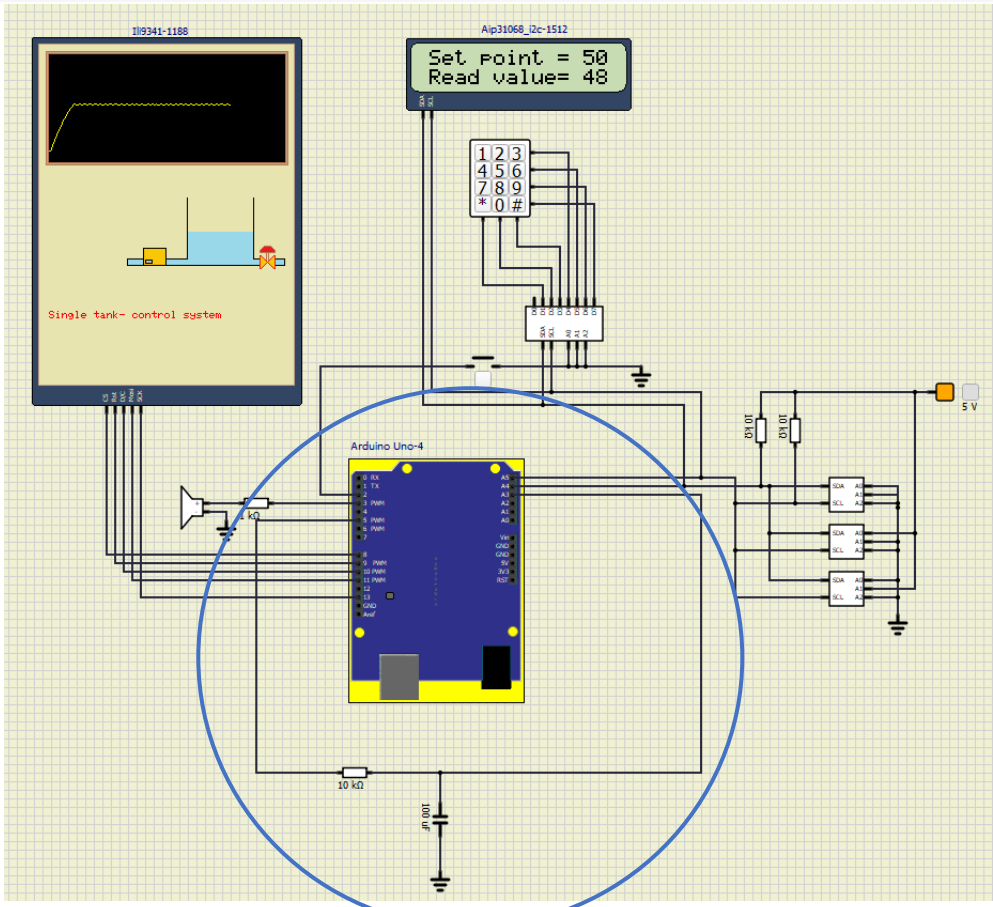


P-Controller for Single Tank system (controller)



P-Controller for Single Tank system (controller)

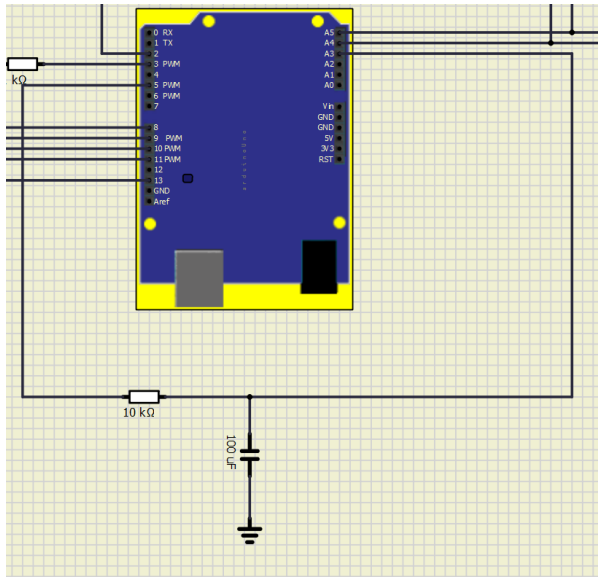




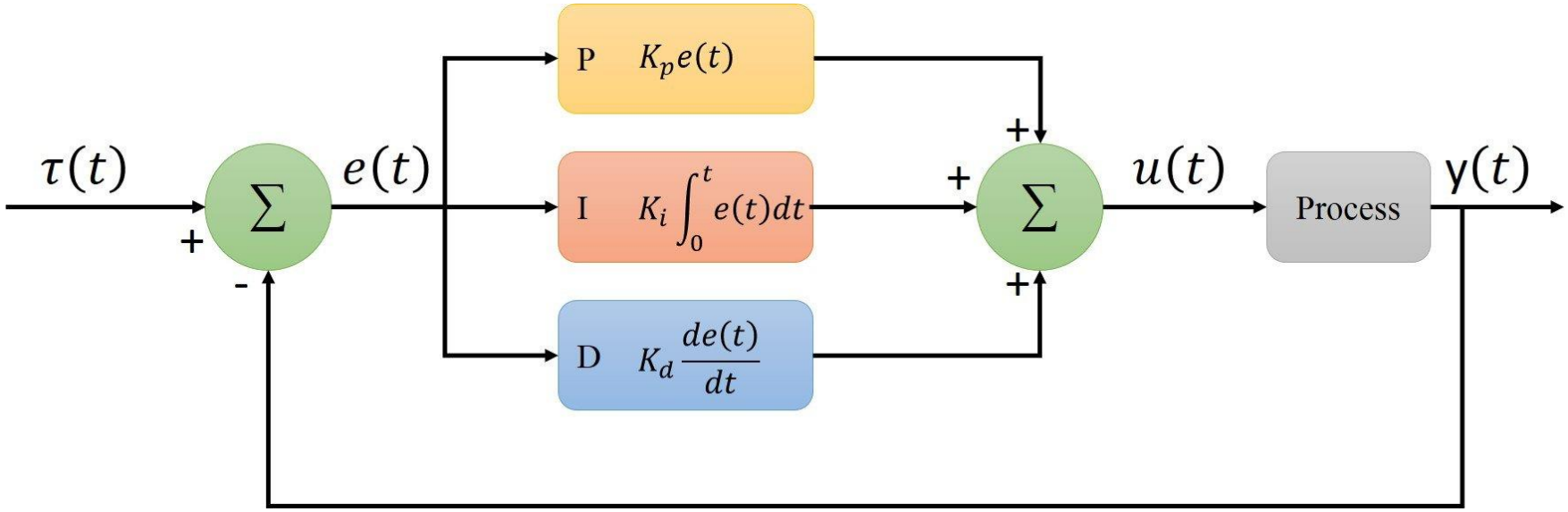
P-Controller for Single Tank system (controller)

```
#define ANALOG_SET_POINT_PIN 2
#define ANALOG_IN_PIN 3
#define ANALOG_OUT_PIN 5
#define SET_POINT_PIN 2
#include <LiquidCrystal_AIP31068_I2C.h>
LiquidCrystal_AIP31068_I2C lcd(62,20,2);
void setup() {
   pinMode(SET_POINT_PIN,INPUT_PULLUP
   lcd.init();
}
float read_val(void){
    int val = analogRead(ANALOG_IN_PIN);
    float ret=100*(val/1023.0);
    return (ret);
}
void set_val(float f_v){
    int i_v=(int)((f_v/100.0)*254);
    i_v=(i_v<0)?1:((i_v>254)?254:i_v);
    analogWrite(ANALOG_OUT_PIN,i_v);
}
int get_set_point(void){
    lcd.setCursor(0,0);     lcd.print("                ");
    lcd.setCursor(0,1);     lcd.print("                ");
    byte key=0;
    int v=0;
    while(key!=20){
        key=get_key();
        if ((key!=20) && (key!=10)) v=10*v+key;
        lcd.setCursor(1,0);
        lcd.print(v);
    }
    v=(v>100)?100:v;
    return v;
}
```

```
float r=50;
void loop(void) {
    float h=read_val();
    float e=r-h;
    float u=50*e;
    lcd.setCursor(1,0);  lcd.print("Set point = "); lcd.print((byte)r);
    lcd.setCursor(1,1);  lcd.print("Read value= "); lcd.print((byte)h);
    set_val(u);
    if (digitalRead(SET_POINT_PIN)==0) r=get_set_point();
    delay(10);
}
```



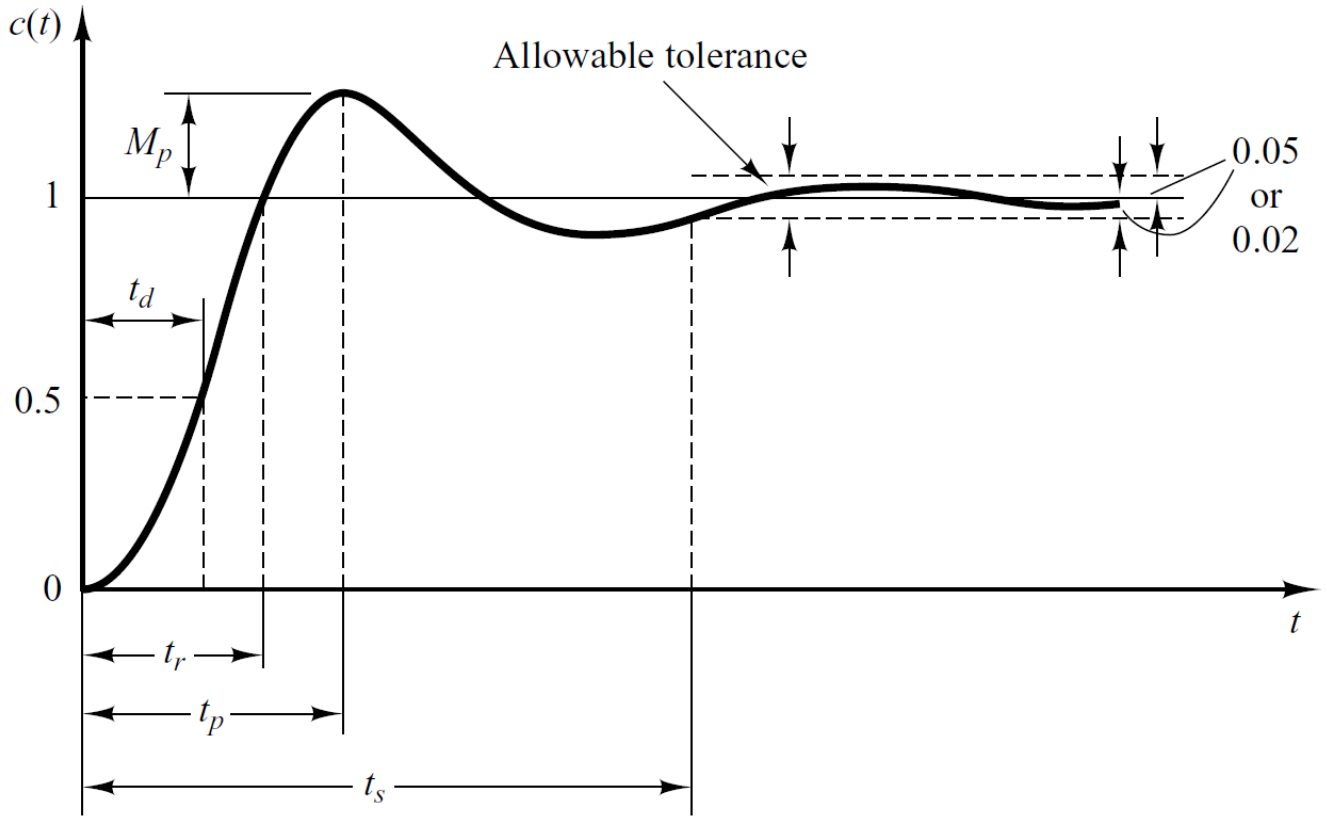
PID-Controller



$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt}$$

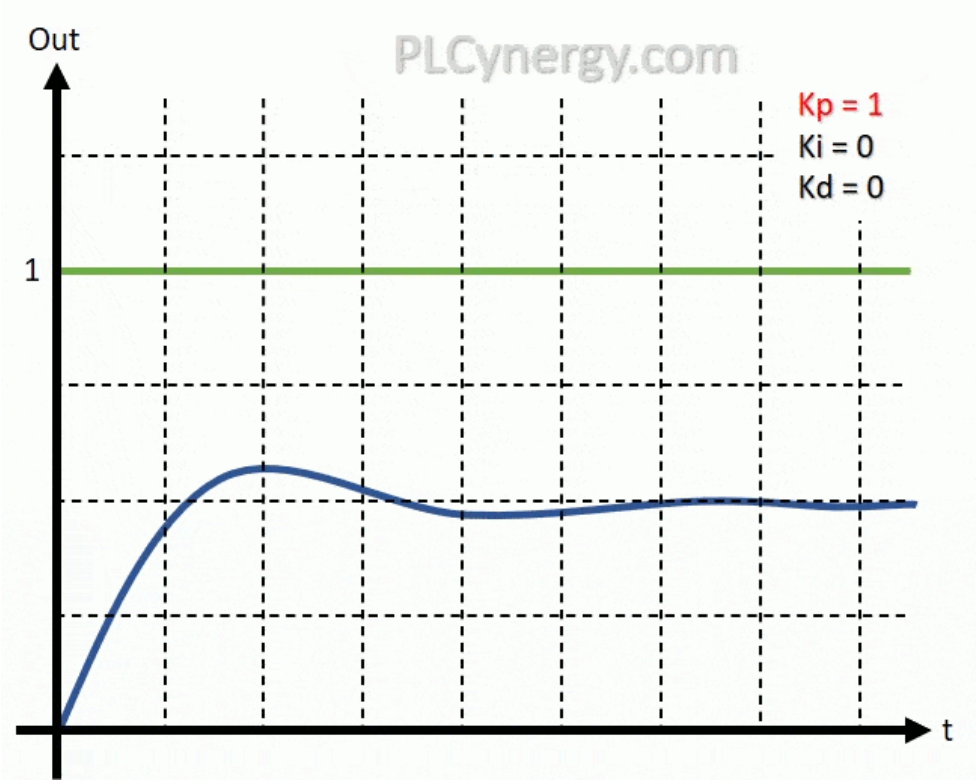
Transient and Steady-State Response Analyses

1) Delay time, t_d
2) Rise time, t_r
3) Peak time, t_p
4) Maximum overshoot, M_p
5) Settling time, t_s



Transient and Steady-State Response Analyses PID(K_p, K_i, K_d) effect

1) Delay time, t_d
2) Rise time, t_r
3) Peak time, t_p
4) Maximum overshoot, M_p
5) Settling time, t_s



Effects of increasing a parameter independently					
Parameter	Rise time	Overshoot	Settling Time	Steady-state Error	Stability
K_p	↓	↑	Small Change	↓	↓
K_i	↓	↑	↑	↓	↓
K_d	Minor Change	↓	↓	Small Change	↓

Transient and Steady-State Response Analyses PID(K_p, K_i, K_d) Manual tuning procedure:

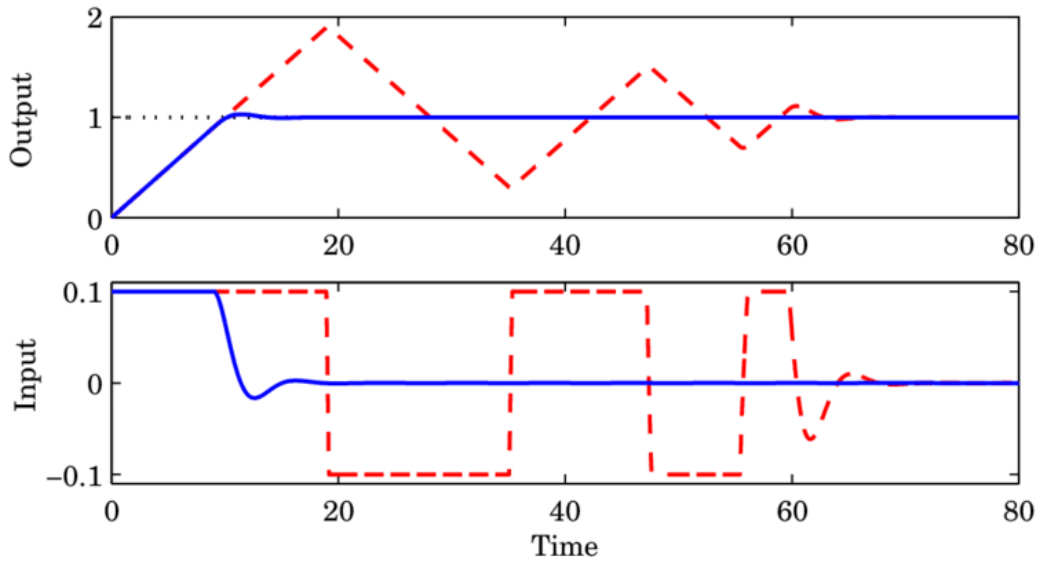
1) Start with a low P gain, usually 1.
2) Increase P until oscillations occur
3) The amplitude of the oscillations is proportional to the error in your system – so it's useful for finding out how good your controller is.
4) Then the P should be set to approximately half of that value
5) Increase I by a small amount and repeat step 2. Keep doing this until you can't find any more improvements
6) Increase D by a small amount and repeat step 2. Keep doing this until you can't find any more improvements and the loop is acceptably quick to reach its reference after a load disturbance

Realization of PID

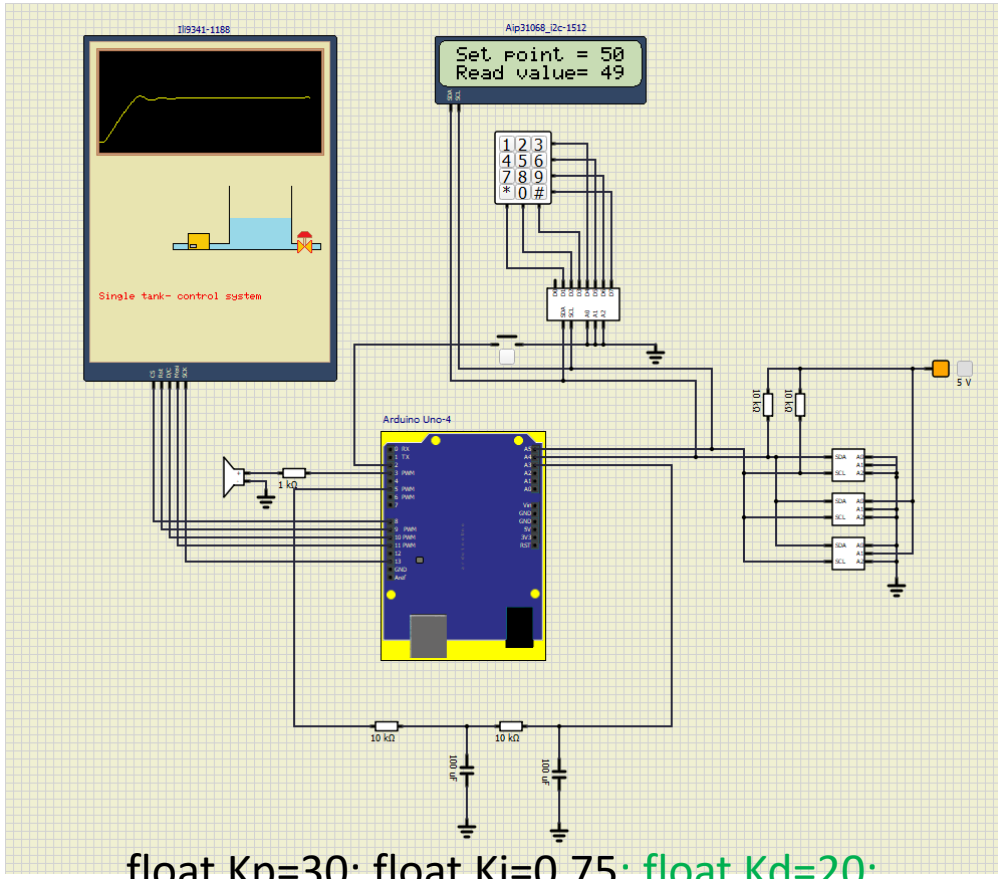
```
213 float r=50;
214 float ui,up,ud;
215 float ui_old;
216 float e;
217 float e_old;
218 float u;
219 float Kp=5; float Ki=0; float Kd=0;
220 void loop(void) {
221     float h=read_val();
222     e=r-h;
223     up=Kp*e;
224     ui=ui+Ki*e;
225     ud=Kd*(e-e_old);
226     u=up+ui+ud;
227     e_old=e;
228     set_val(u);
229     delay(10);
230 }
```

Realization of PID With (Anti-windup)

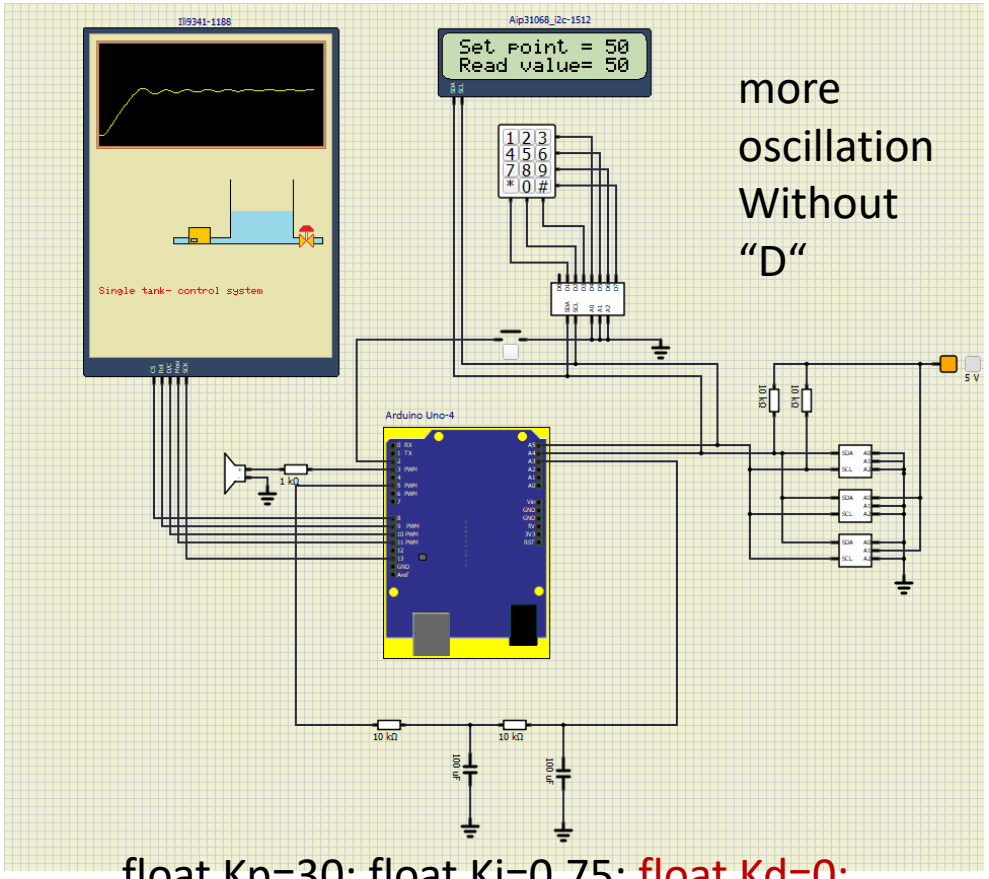
```
213  float r=50;
214  float ui,up,ud;
215  float ui_old;
216  float e;
217  float e_old;
218  float u;
219  float Kp=30; float Ki=0.75; float Kd=20;
220  #define MAX_U 100
221  #define MIN_U 0
222  void loop(void) {
223      float h=read_val();
224      e=r-h;
225      up=Kp*e;
226      ui=ui+Ki*e;
227      ud=Kd*(e-e_old);
228      u=up+ui+ud;
229      if ( (u>MAX_U) || (MIN_U<0) ) {
230          ui=ui_old;
231          u=up+ui+ud;
232      }
233      e_old=e;
234      ui_old=ui;
235      set_val(u);
236      if (digitalRead(SET_POINT_PIN)==0)  r=get_set_point();
237      delay(10);
238  }
```



Waveform response on SimullDE for PID with Double Tank System



float Kp=30; float Ki=0.75; float Kd=20;



more oscillation Without "D"

float Kp=30; float Ki=0.75; float Kd=0;

References

1) https://www.amazon.com/Modern-Control-Engineering-Katsuhiko-Ogata/dp/0136156738
2) https://www.researchgate.net/figure/Bit-rearrangement-substep-applied-to-every-pixel-of-a-source-image-for-obtaining_fig5_333883036
3) https://plcynergy.com/pid-controller/
4) https://www.researchgate.net/figure/Illustration-of-integrator-windup-The-dashed-red-lines-show-the-response-with-an-ordinary_fig11_267198441