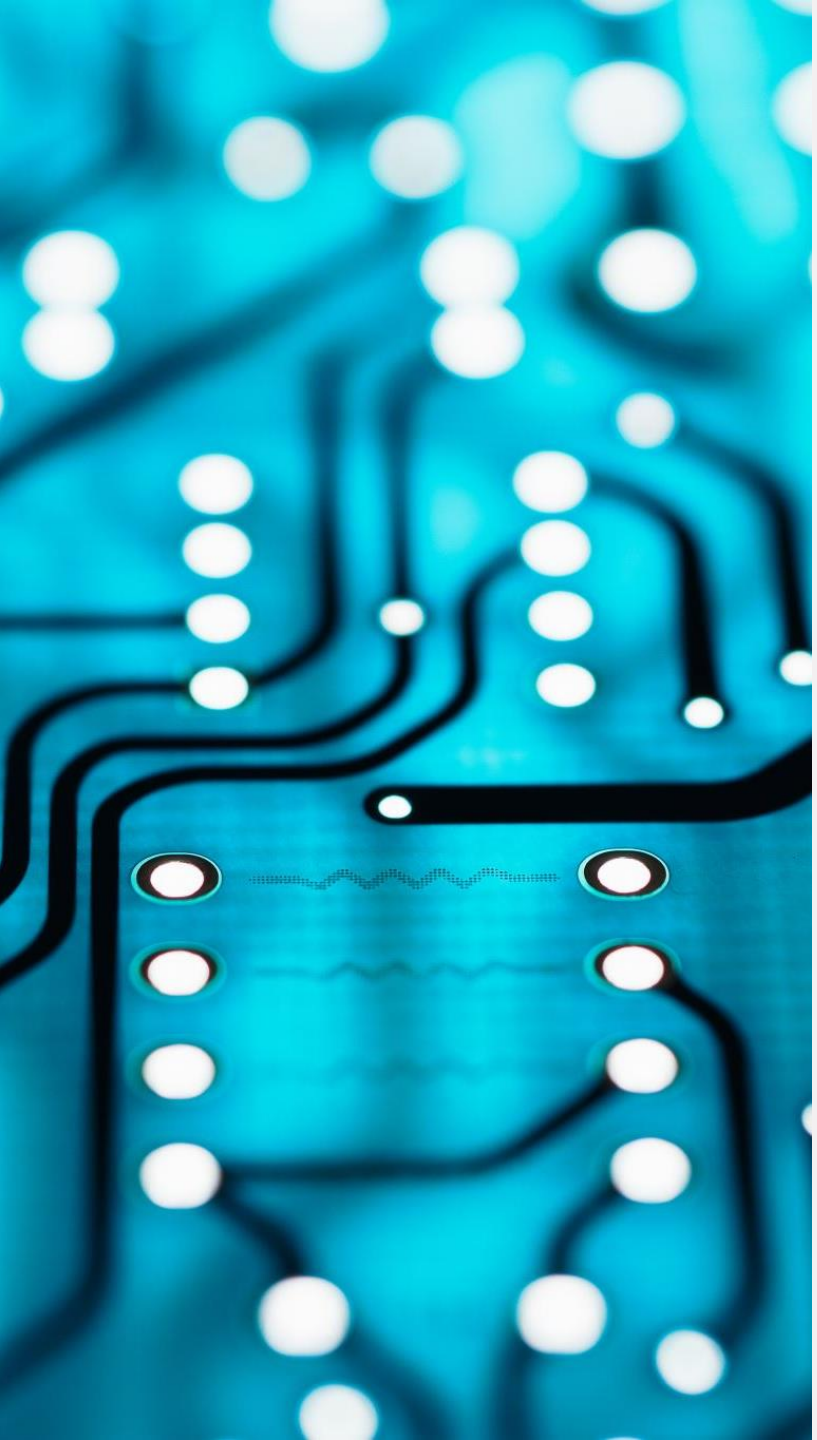


Embedded Systems (EPM)

Lecture (10) Summary

Mathematical Mode of a single tank

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

Q = H/R , same as I = V/R

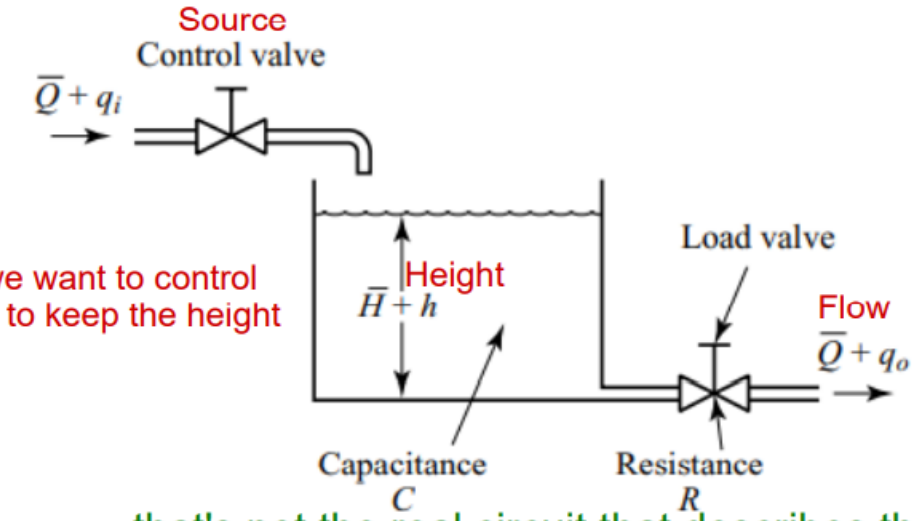
RC is the time constant that refers to the time that tank will full

If R=1

Output / Input

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

After Laplace transform



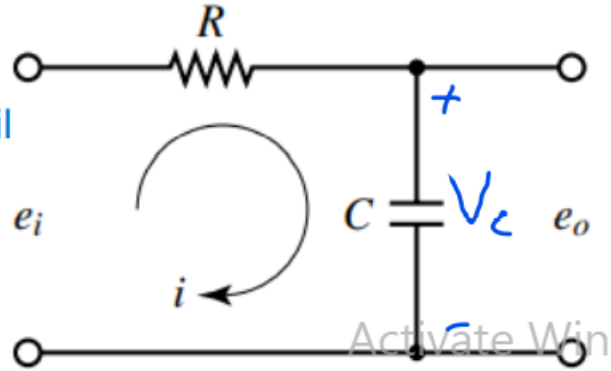
The Tank we want to control as we want to keep the height constant.

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

If R=1

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

that's not the real circuit that describes the tank

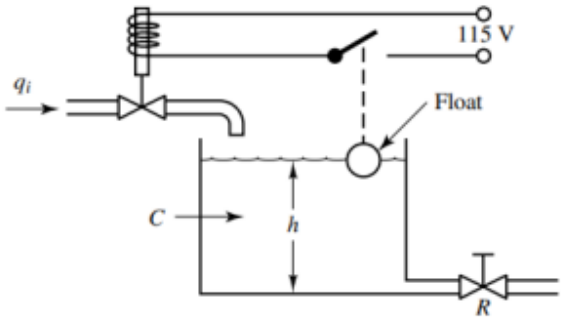


we charge the capacitor here so Vc increases until it equals ei so there's no current passes so Static gain = 1

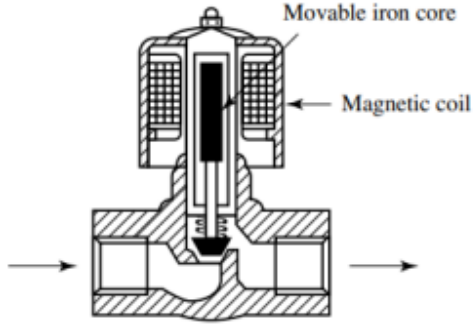
We can consider

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

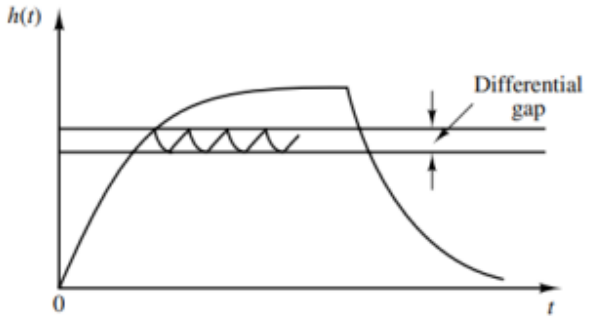
On-off controller for single tank



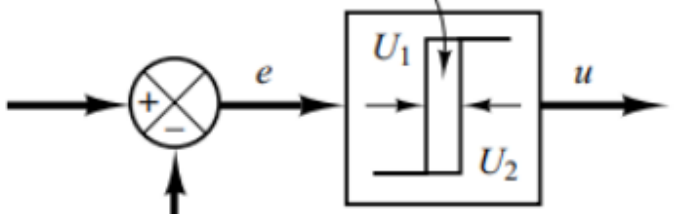
(a)



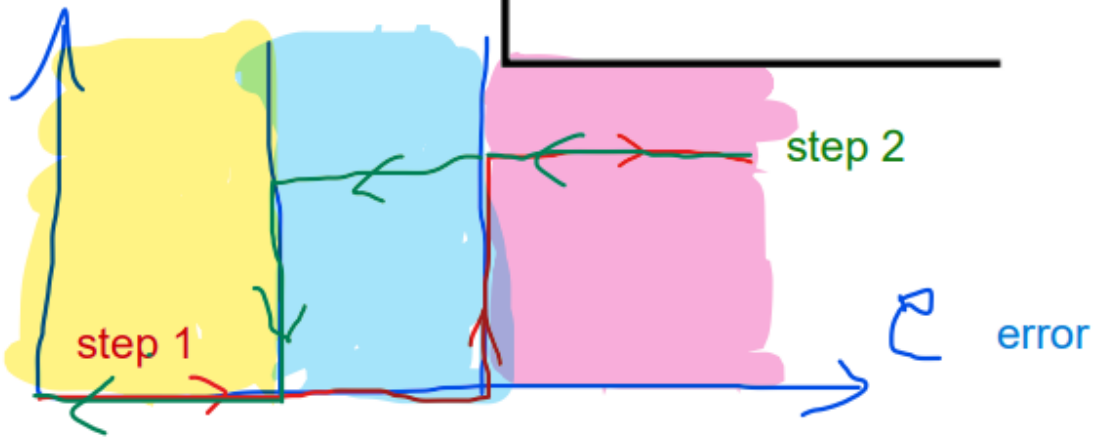
(b)



when its area becomes very small the system may be crashed although don't make this area very large to make the system more efficient



volt of pump

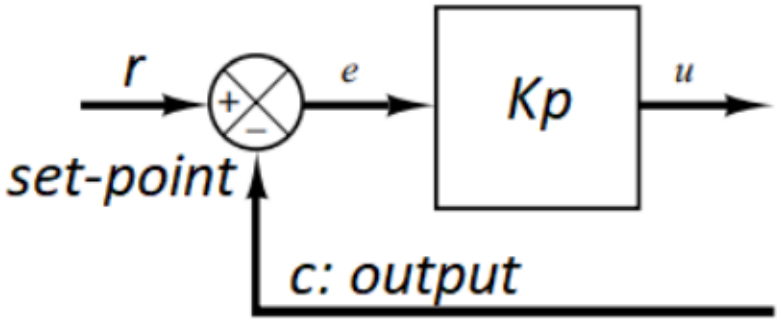


Proportional Control Action

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

Control action is Proportional to the error value

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



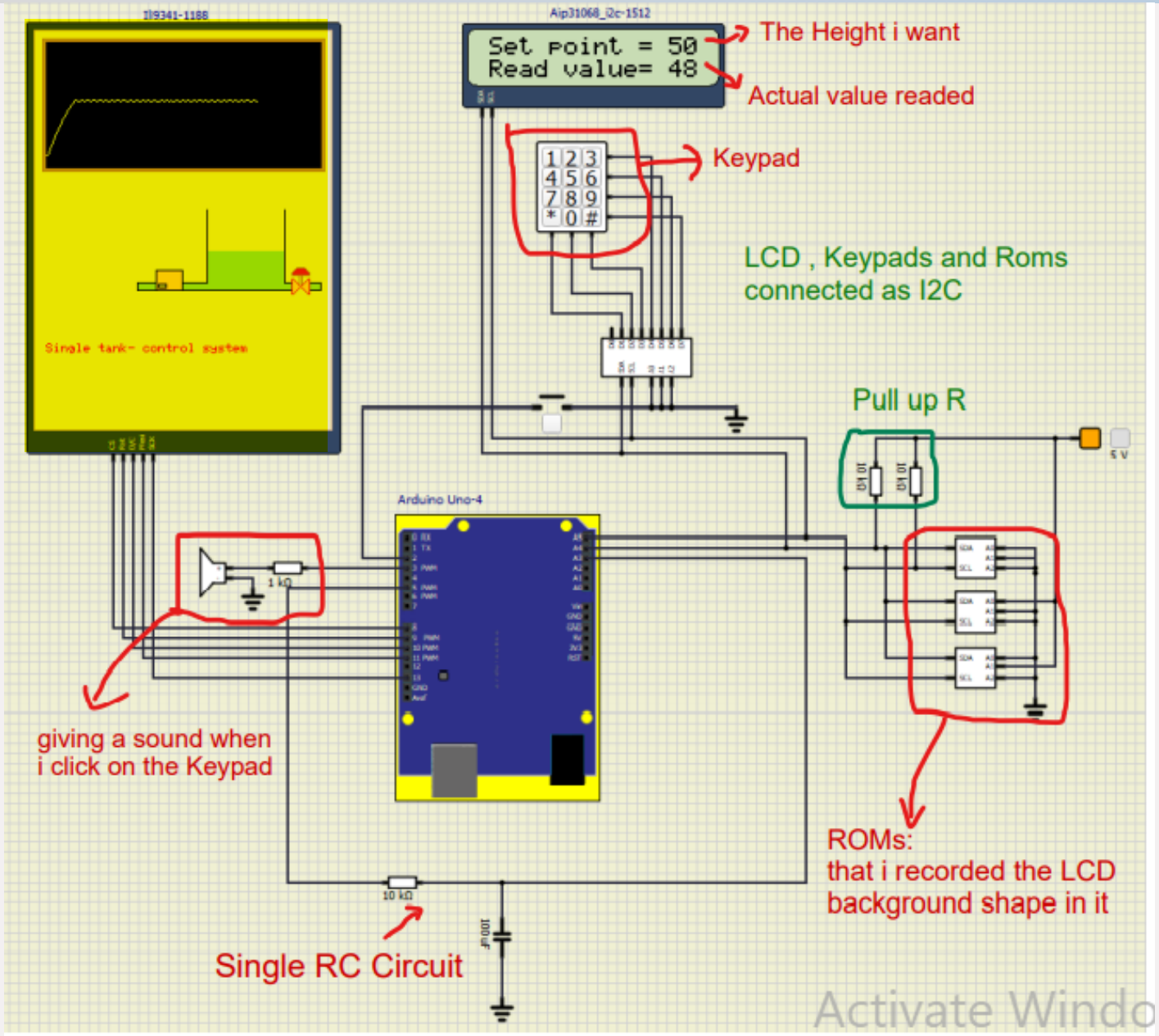
or, in Laplace-transformed quantities,

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

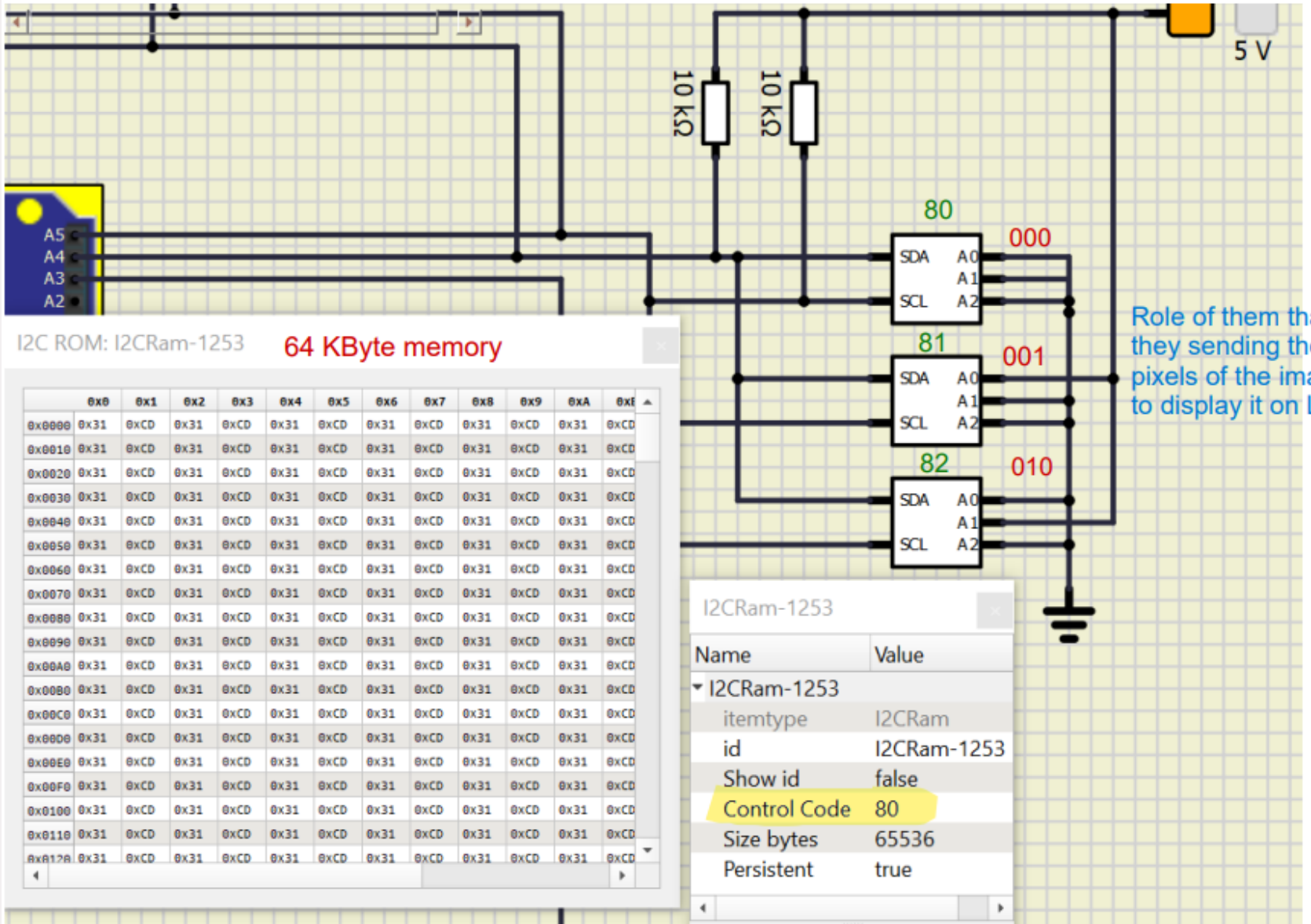
where K_p is termed the proportional gain.

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.

P-Controller for Single Tank system



EEPROM/ROM



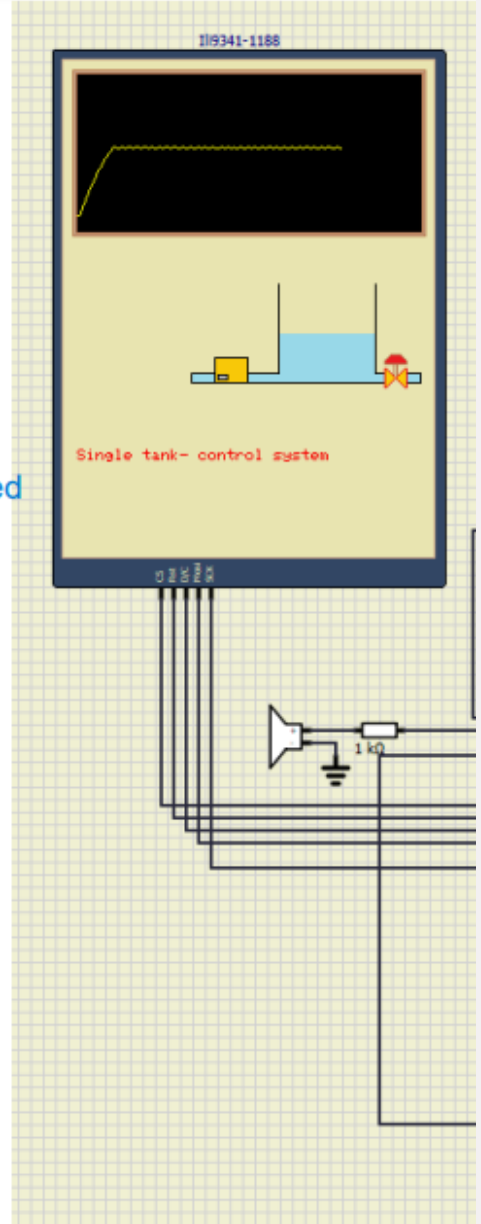
I2C ROM: I2CRam-1253 — 64 KByte memory

	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB
0x0000	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD
0x0010	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD
0x0020	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD
0x0030	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD
0x0040	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD
0x0050	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD
0x0060	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD
0x0070	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD
0x0080	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD
0x0090	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD
0x00A0	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD
0x00B0	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD
0x00C0	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD
0x00D0	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD
0x00E0	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD
0x00F0	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD
0x0100	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD
0x0110	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD
0x0120	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD	0x31	0xCD

I2CRam-1253

Name	Value
I2CRam-1253	
itemtype	I2CRam
id	I2CRam-1253
Show id	false
Control Code	80
Size bytes	65536
Persistent	true

Role of them that they sending the stored pixels of the image to display it on LCD



P-Controller for Single Tank system

EEPROM/ROM Code

Read by Order of Numbers to be more simple

```
#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;
        }
    }
```

5 - sending pointer(address) of Array to full it

7 - masking address with 16 bits so all the number will become zeros then shift Right 16 bits so i can see the 2 bits which are (00,01,10) that determine the memory address that i will take data from

6- if the last 2 bits is 00 then Read from memory 80
if 01 then Read from memory 81
if 10 then Read from memory 82

1-MEMRead() function that read from memory starting from (address) and read from array(data)16 location

4- array of 16 element

2- at First address equals 0

3 - Every time address value increases by 32 (2*NUM_OF_WORD = 2*16)

```
#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();
}
```

8- MEM_ID = 80 + (0 or 1 or 2,according to Rom Address)

9- address anding with 8 Most sig. bits then shifting them 8 bits to Right

10 - anding address with 8 Lest Sig. bits

11 - Reading from the selected memory 32 Bytes(16*2)

LSB

MSB

12 - shifting read_val2 8 bits to Right to be MSB and adding them

TFT Screen/SPI

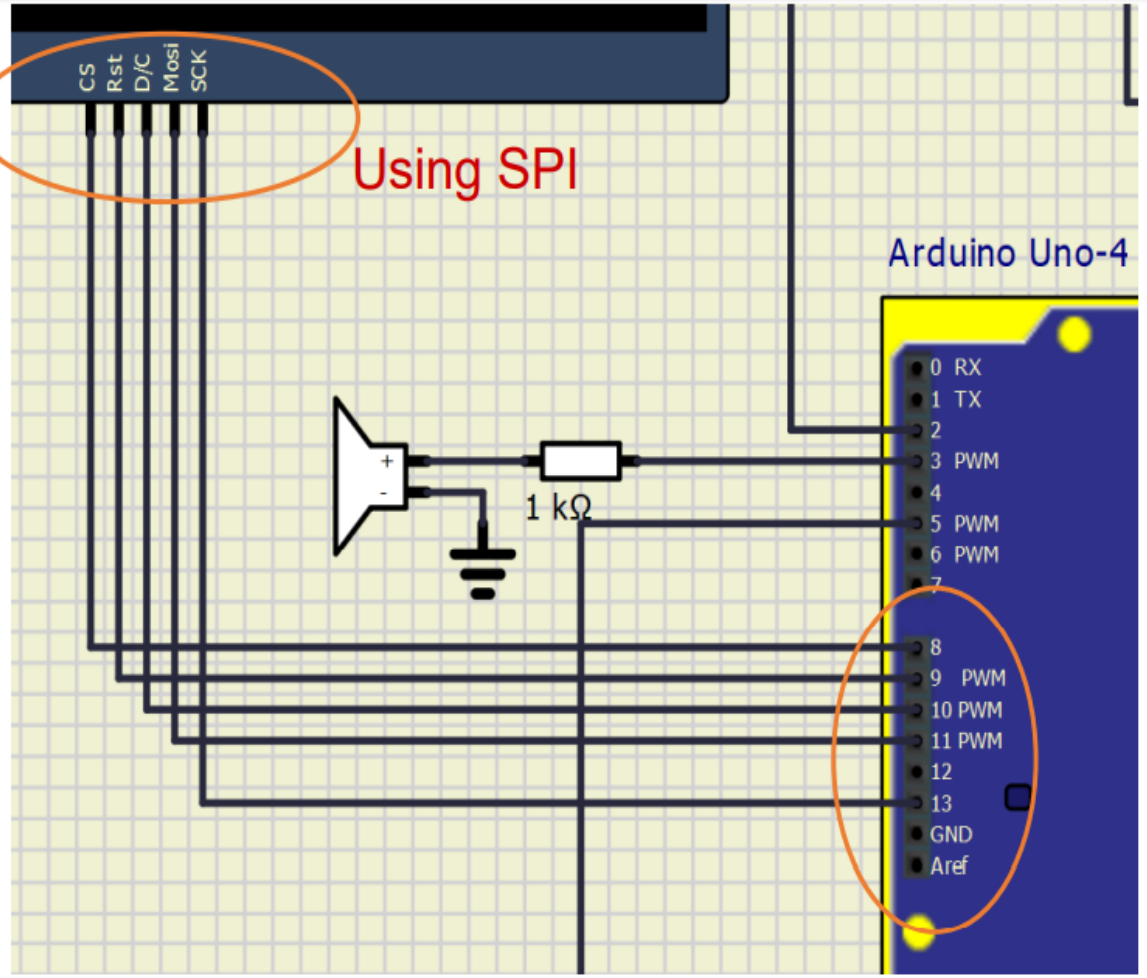
```
#include <Adafruit_ILI9341.h>
#define TFT_CS   8
#define TFT_RST  9
#define TFT_DC   10
#define TFT_MOSI 11
#define TFT_MISO 12  // not used
#define TFT_CLK  13

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;
}
```

we must connect to this pins
MOSI 11 , MISO 12 , CLK 13 because here we use Hardware SPI

Reading data and draw them

tft.drawline(x1,y1,x2,y2,the color)



Keypad

```ino
#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;
}
```

D7 D6 D5 D4 D3 D2 D1 D0

0xfd = 1111 - 1101

0xfb = 1111 - 1011

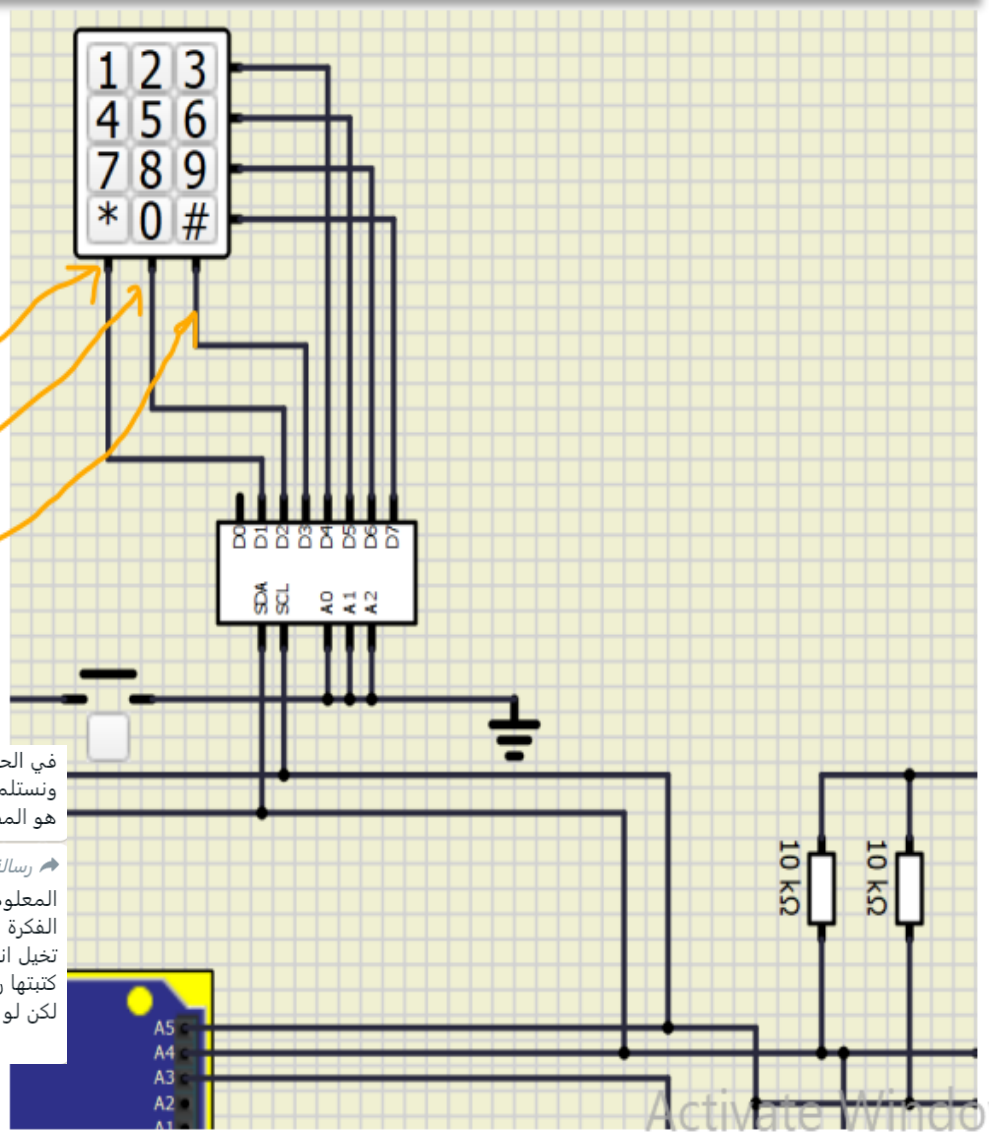
0xf7 = 1111 - 0111

took the number shift it and invert it

في الحتة الي الي بنشيل ايدينا من على الزرار دي انا كنت قولت بنكتب اي رقم عشوائي ونستلمه تاني
هو المفروض لازم نكتب 0x0f او 0xf0 مش اي رقم تاني
١:٣٩ م

رسالة محوّلة

المعلومة الي فوق كافية عادي كبصمجة يعني بس لو حد عايز يفهم ليه
الفكرة انك لو كاتب مثلا اي رقم عشوائي زي 0110 1111
تخيل انك دوست على زرار خلاك توصل ال 1ب ال 1 فانت كدا هتقرا نفس القيمة الي كتبتها رغم من انك لسا دايس عل زرار
لكن لو ماتب 0000 1111 اي توصليها هتخلي الواحد يبقى زيرو او الزيرو يبقى واحد
١:٣٩ م



(controller)

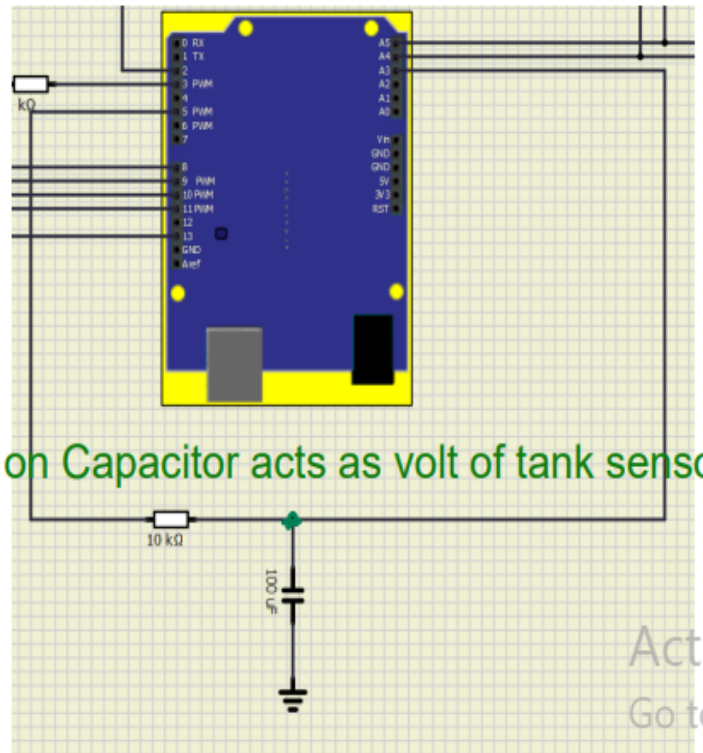
```
#define ANALOG_SET_POINT_PIN 2
#define ANALOG_IN_PIN 3
#define ANALOG_OUT_PIN 5
#define SET_POINT_PIN 2
#include <LiquidCrystal_AIP31068_I2C.h>
LiquidCrystal_AIP31068_I2C lcd(62,20,2);
void setup() {
   pinMode(SET_POINT_PIN,INPUT_PULLUP
   lcd.init();
}
float read_val(void){
    int val = analogRead(ANALOG_IN_PIN); Read Value
    float ret=100*(val/1023.0); 1023.0 to cast it as a float
    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;
}
```

take the amplified error and map it to Arduino

20 (#) acts a Enter Key for Keypad

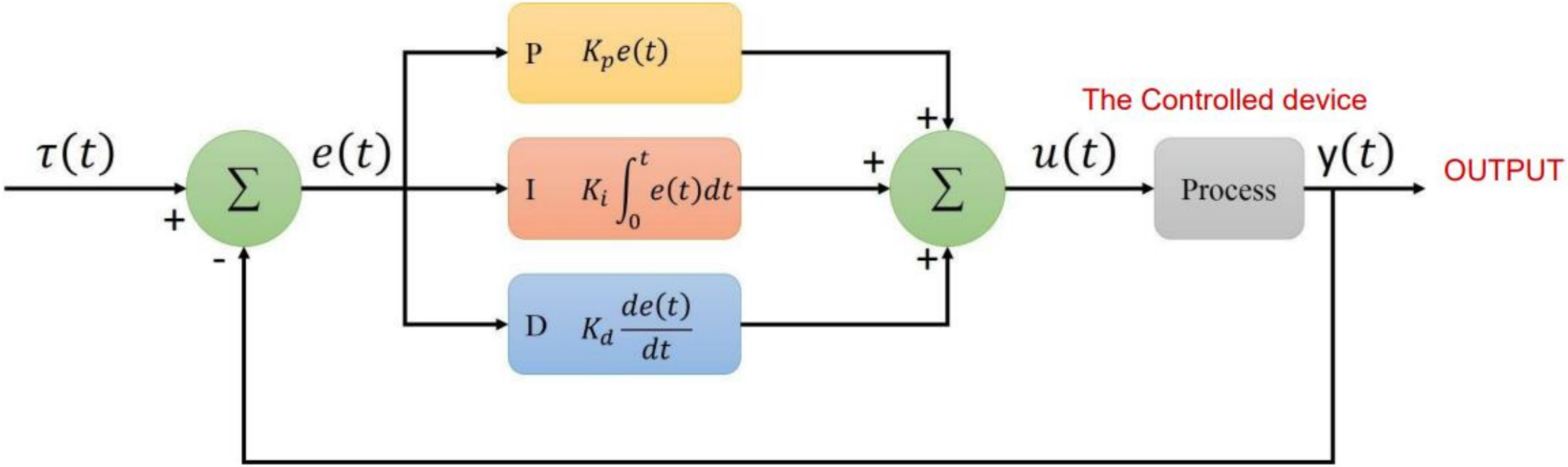
if the user enters a number >100
this condition makes maximum number equals 100

```
float r=50;
void loop(void) {
    float h=read_val(); Reads water height
    float e=r-h;        Calculate Error
    float u=50*e;       multip. error by 50
    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);
}
```



Volt on Capacitor acts as volt of tank sensor

PID-Controller

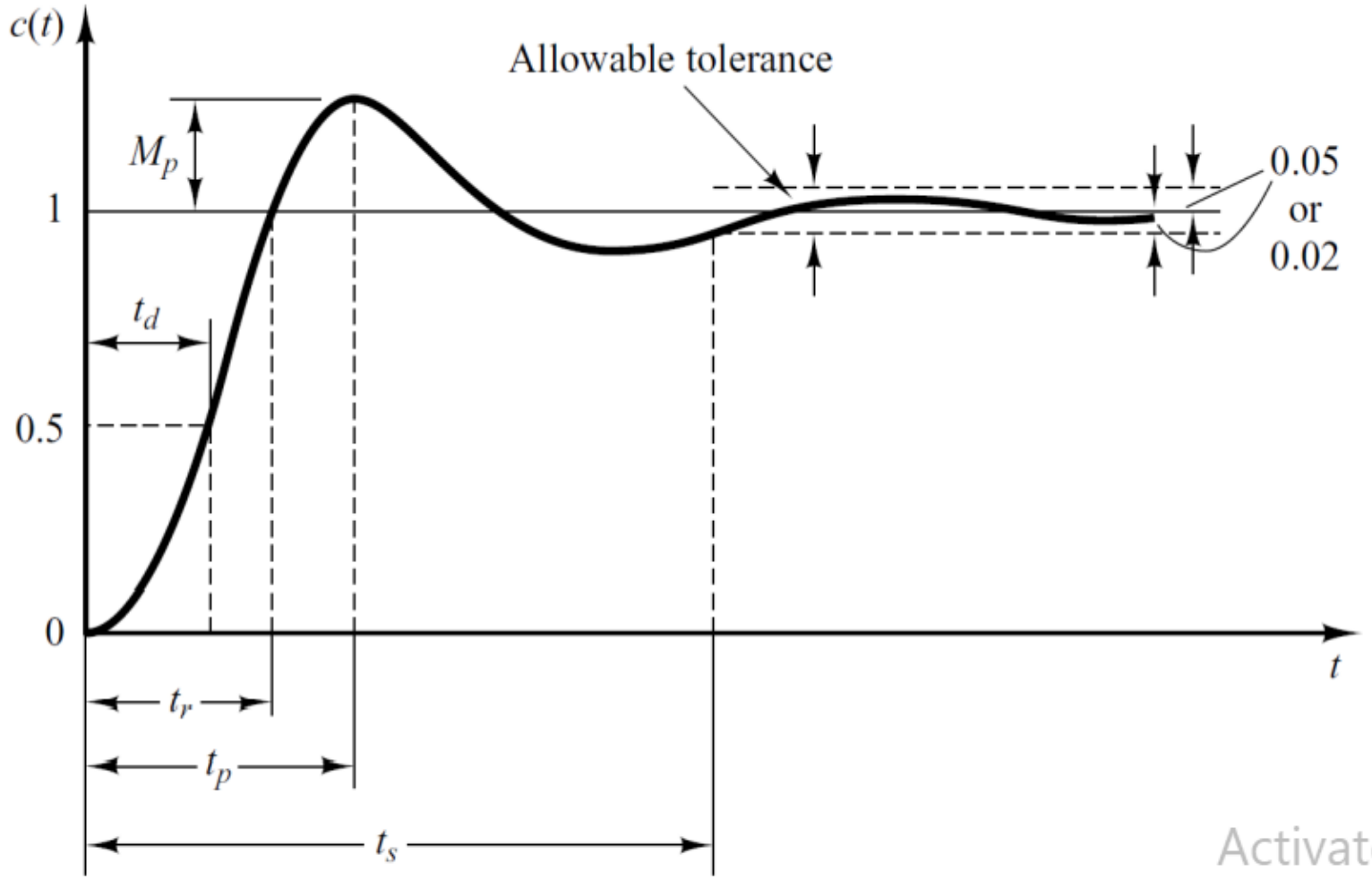


$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt}$$

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

System is more better when all this factors are small

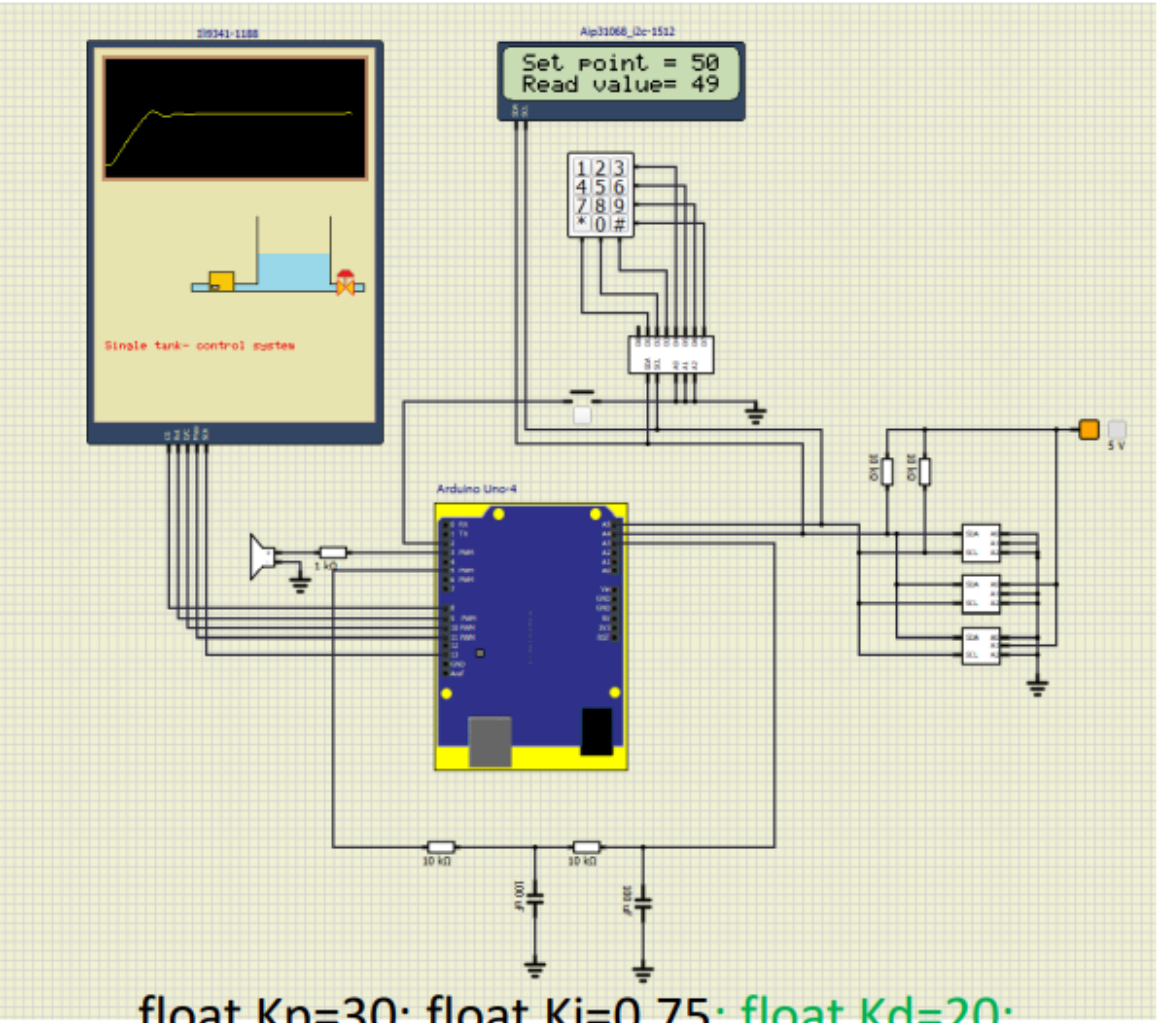


Effects of increasing a parameter independently					
Parameter	Rise time	Overshoot	Settling Time	Steady-state Error	Stability
↑ K_p	↓	↑	Small Change	↓	↓
↑ K_i	↓	↑	↑	↓	↓
↑ K_d	Minor Change	↓	↓	Small Change	↓

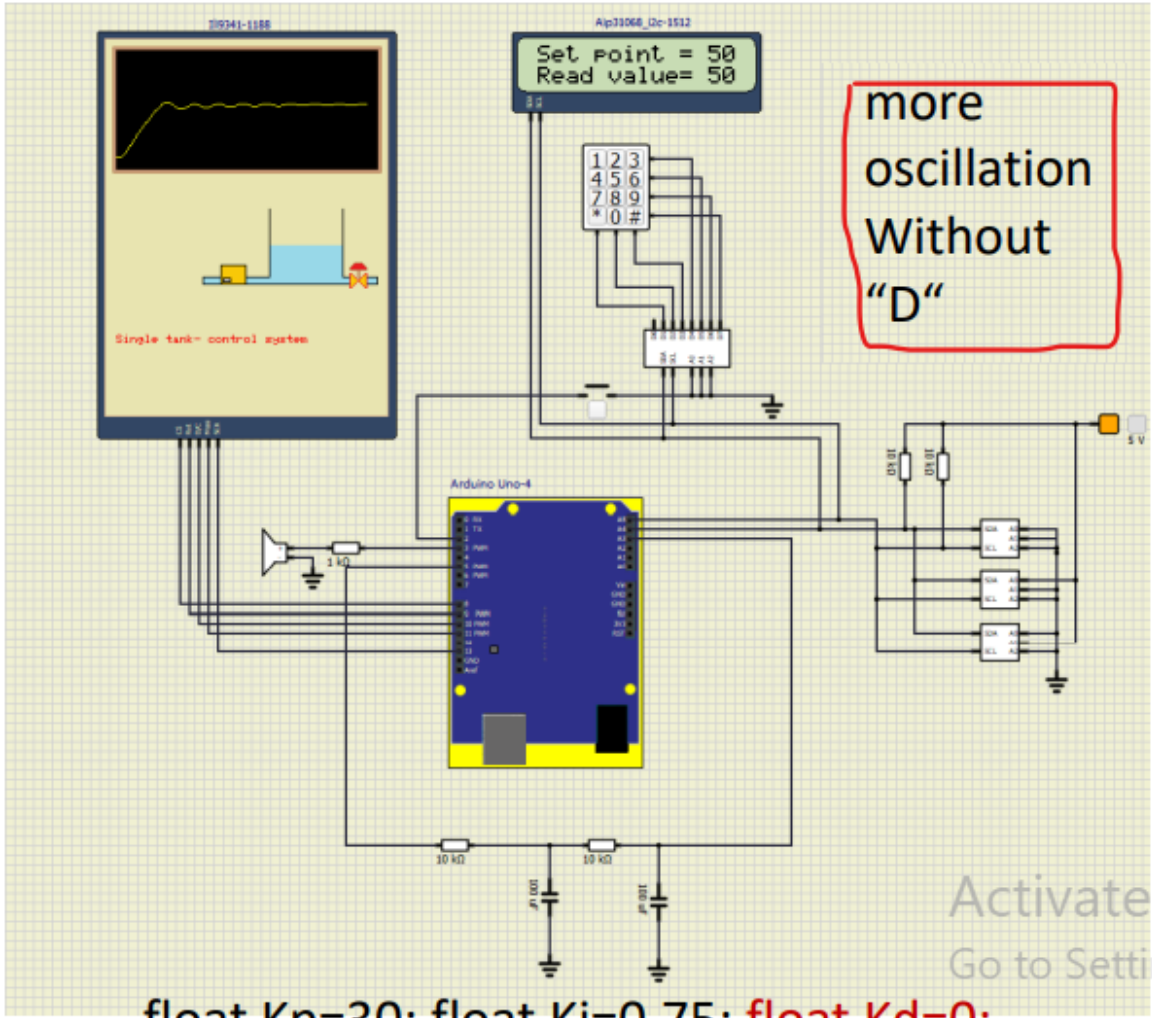
Transient and Steady-State Response Analyses PID(K_p, K_i, K_d) Manual tuning procedure:

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

Waveform response on SimullDE for PID with Double Tank System



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



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