BLG212 Hw 4

Mehmet Şerbetçioğlu
040160056

$f = 210$ Hz

Tasarlanan IIR Filtresinde cutofflar → $f_1 = 205$ Hz , $f_2 = 215$ Hz

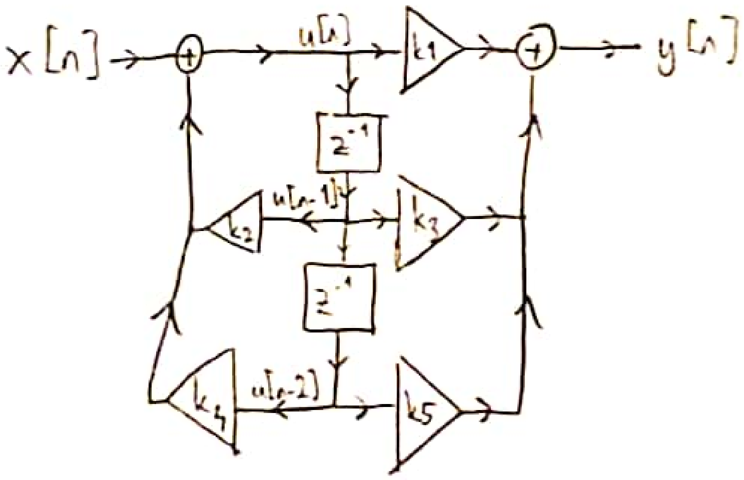
örnekleme frekansı → $F_s = 4800$ Hz
(baudrate)

2. derece Butterworth filtresi:

$$H(s) = \frac{64.03\,s}{s^2 + 64.03s + 1{,}762.10^6}$$

$s = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}$ dönüşümü yapıldığında;

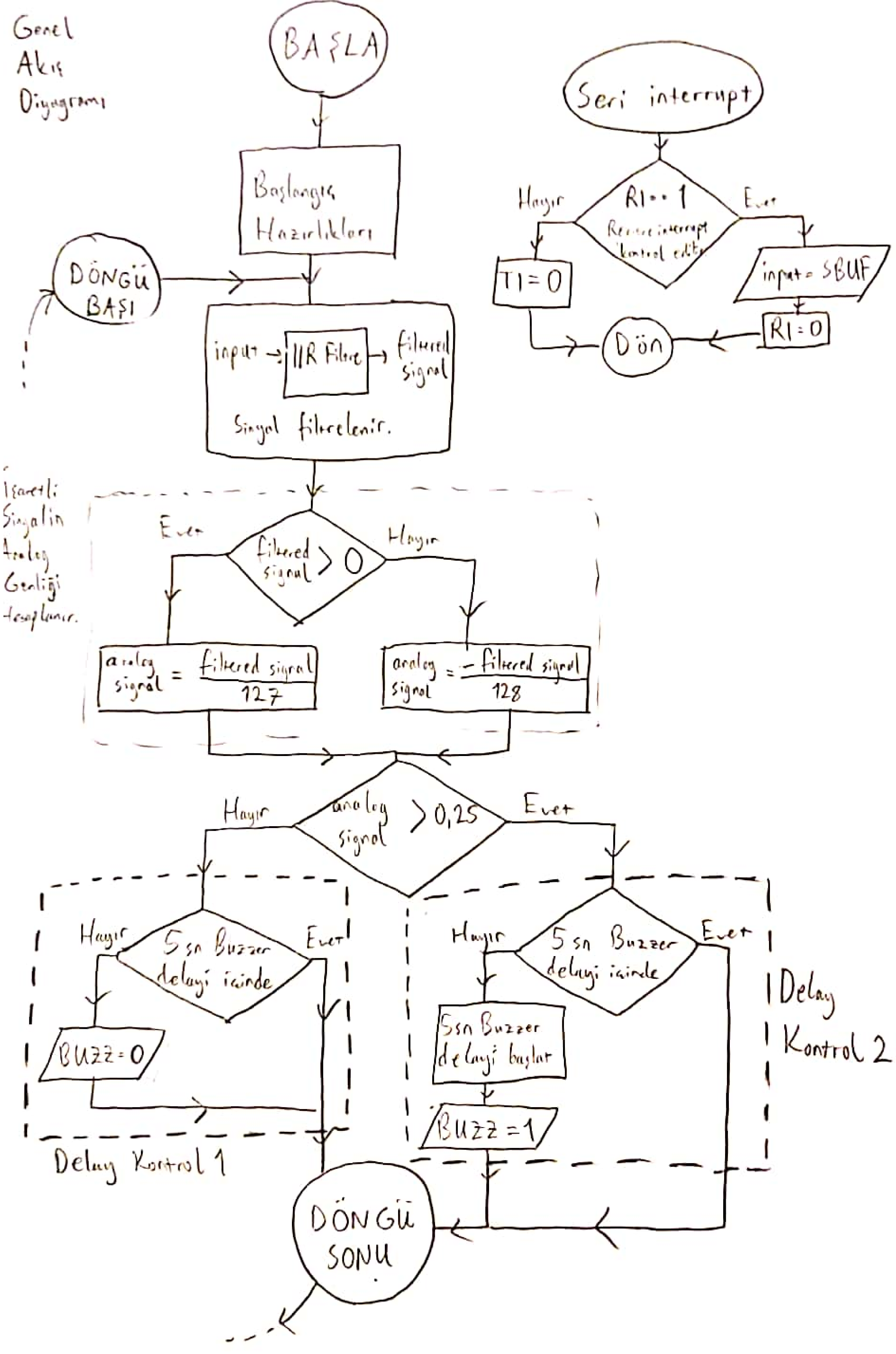
$$H(z^{-1}) = \frac{0{,}006502 - 0{,}006502\,z^{-2}}{1 - 1{,}912443\,z^{-1} + 0{,}986996\,z^{-2}} = \frac{Y(z^{-1})}{X(z^{-1})}$$



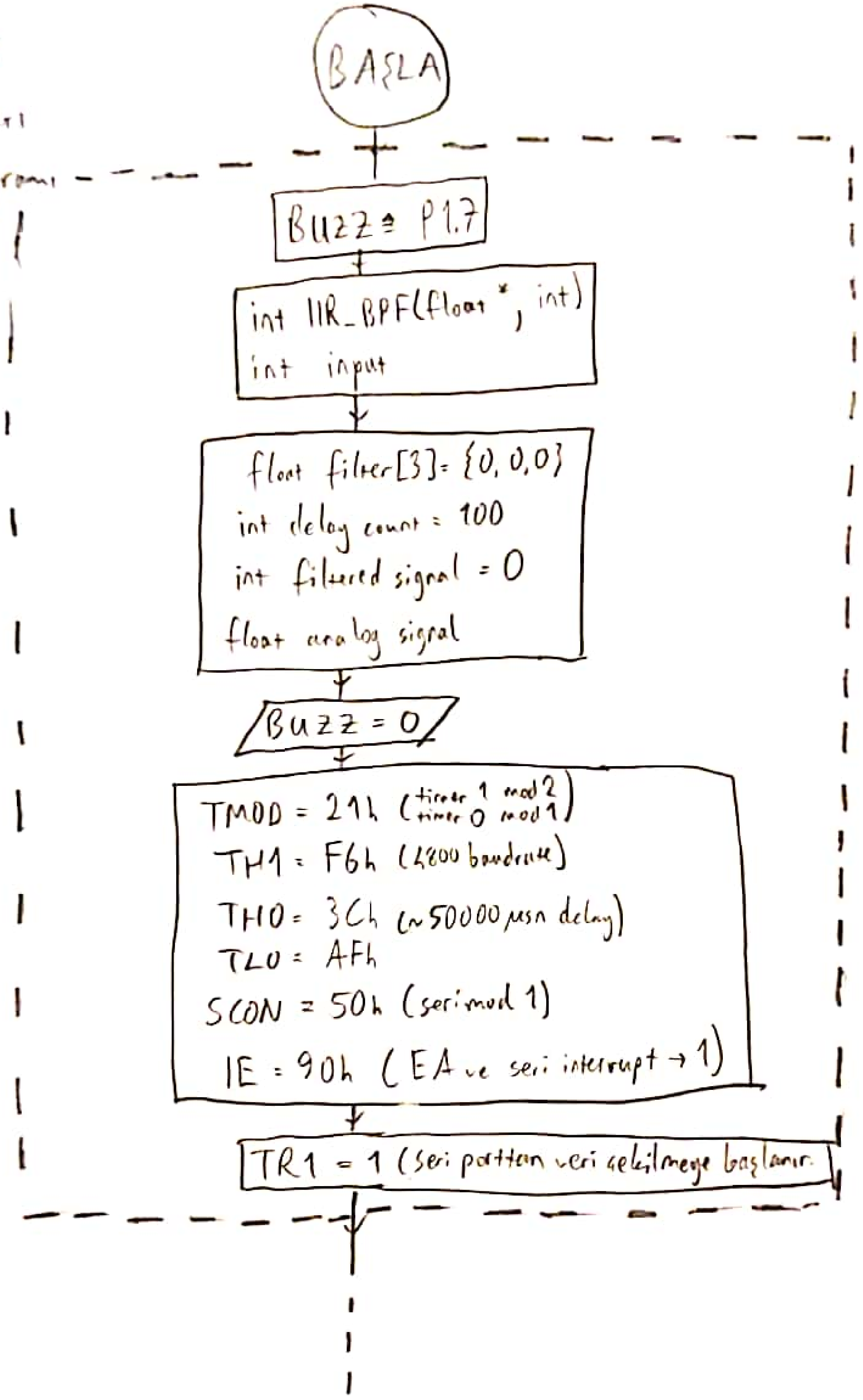
$u[n] = x[n] + 1{,}912443\,u[n-1] - 0{,}986996\,u[n-2]$

$y[n] = 0{,}006502\,u[n] - 0{,}006502\,u[n-2]$

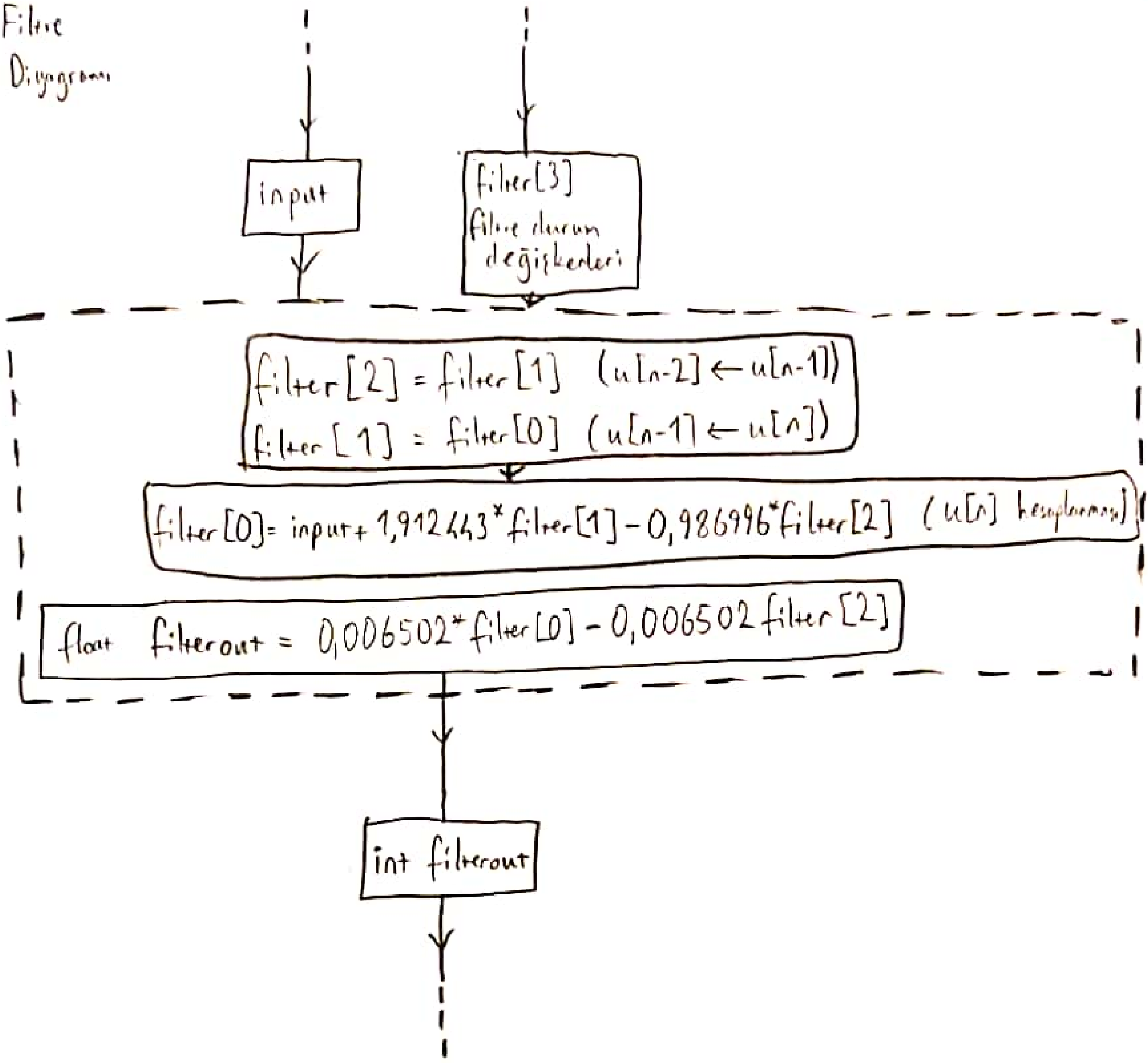
Genel Akış Diyagramı



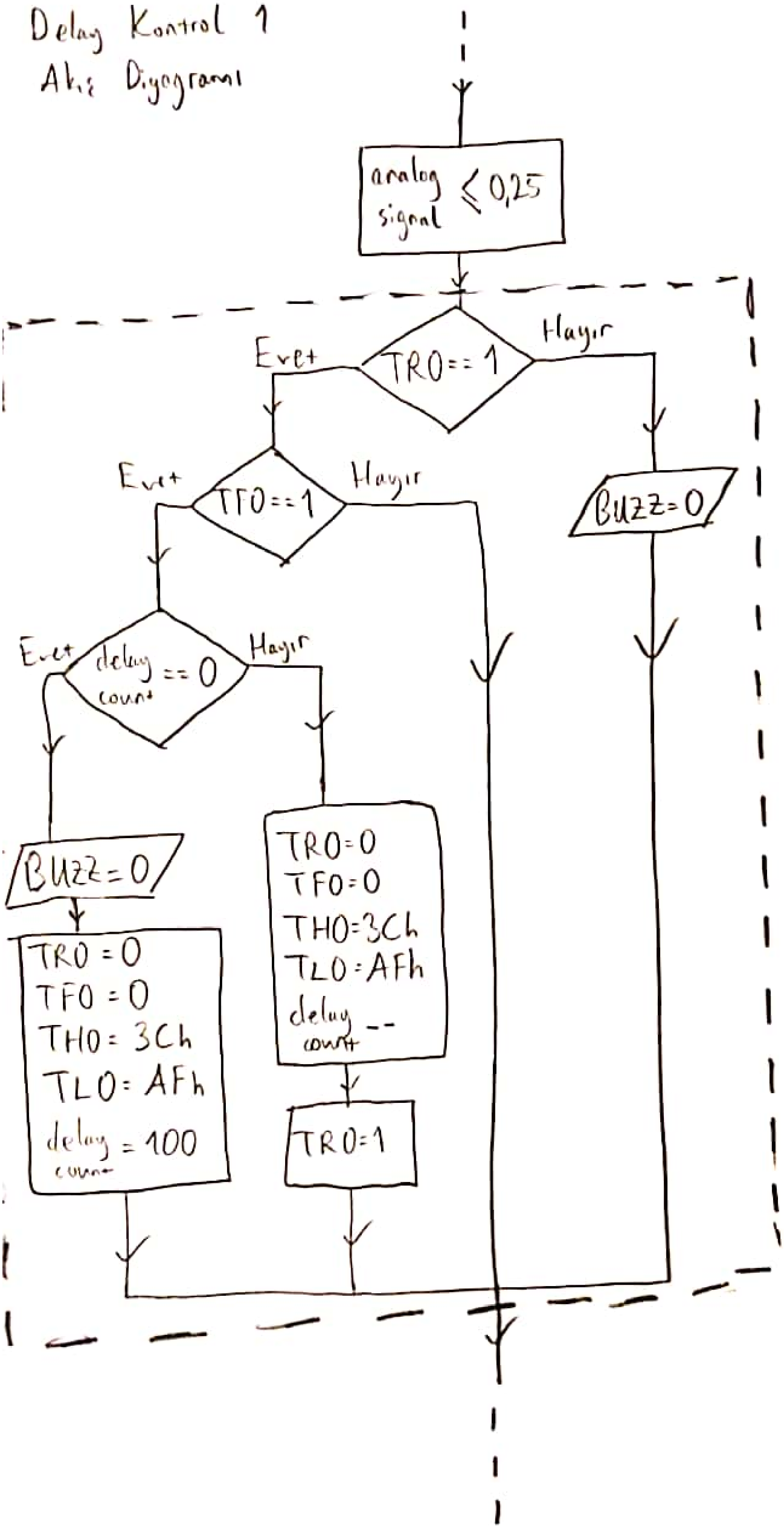
Başlangıç Hazırlıkları Akış Diyagramı


BAŞLA
BUZZ ≙ P1.7
int IIR_BPF(float *, int)
int input
float filter[3]= {0, 0,0}
int delay count = 100
int filtered signal = 0
float analog signal
BUZZ = 0
TMOD = 21h (timer 1 mod 2 / timer 0 mod 1)
TH1 = F6h (4800 baudrate)
TH0 = 3Ch (~50000 μsn delay)
TL0 = AFh
SCON = 50h (seri mod 1)
IE = 90h (EA ve seri interrupt → 1)
TR1 = 1 (Seri porttan veri çekilmeye başlanır.

IIR Filtre
Akış Diyagramı

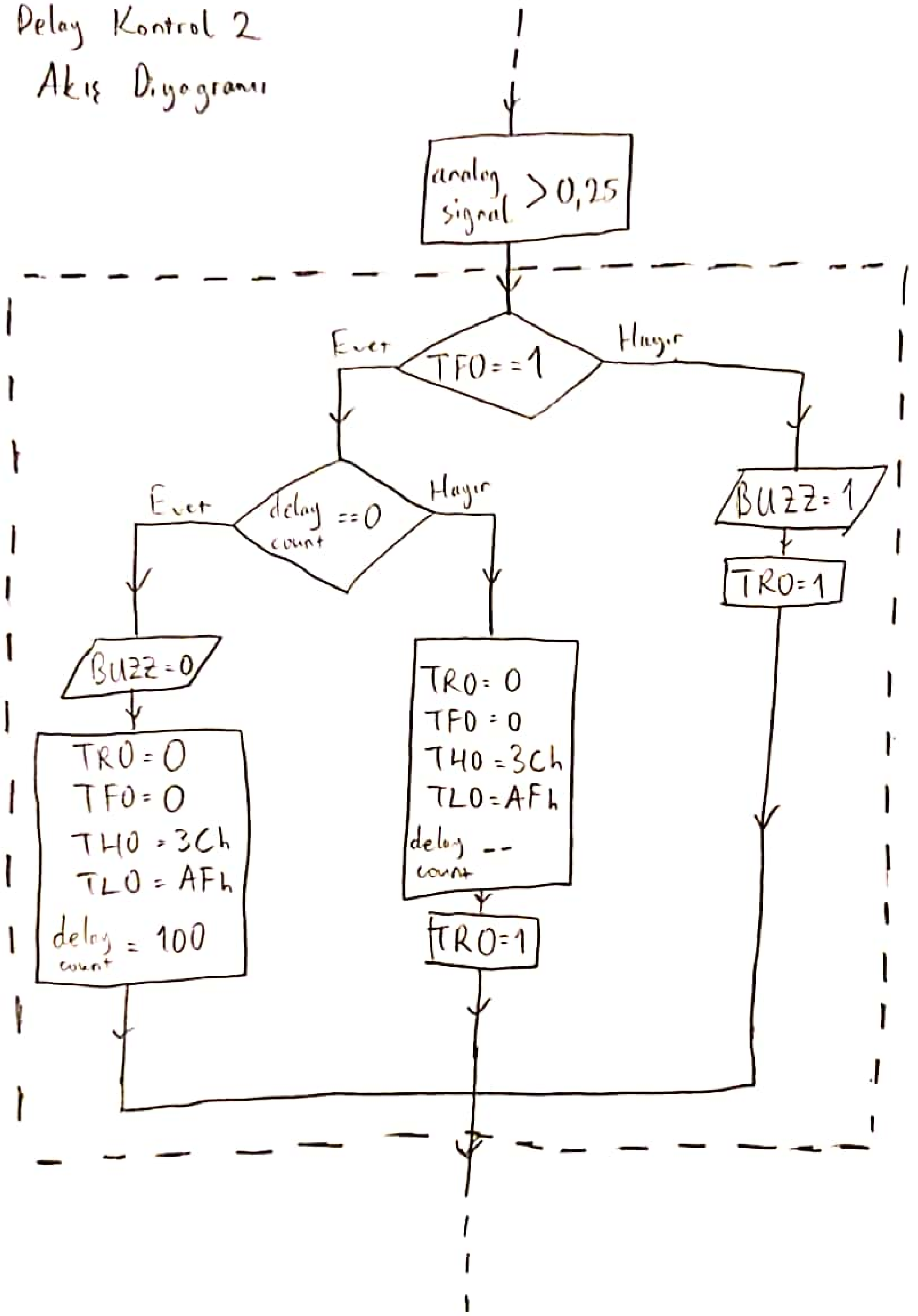

input
filter[3]
filtre durum değişkenleri
filter[2] = filter[1] (u[n-2] ← u[n-1])
filter[1] = filter[0] (u[n-1] ← u[n])
filter[0]= input+ 1,912443*filter[1]-0,986996*filter[2] (u[n] hesaplanması)
float filterout = 0,006502*filter[0] - 0,006502 filter[2]
int filterout

Delay Kontrol 1
Akış Diyagramı



CamScanner ile tarandı

Delay Kontrol 2
Akış Diyagramı



C kodu:

```
#include <reg51.h>
sbit BUZZ= P1^7;
int IIR_BPF(float *, int);
int input;
void serial0() interrupt 1
{
    if (R1==1)
    {
        input = SBUF;
        R1 = 0;
    }
    else
    {
        T1=0;
    }
}
void main(void)
{
    float filter[3]= {0};
    int delay_count = 100;
    int filtered_signal = 0;
    float analog_signal;
    BUZZ=0
    TMOD= 0x21;
    TH1= 0xF6;
    TH0= 0x3C;
    TL0= 0xAF;
    SCON= 0x50;
    IE= 0x90;
    TR1=0
```

```
while (1)
{
 filtered_signal = IIR_BPF(filter, input);
 if (filtered_signal > 0)
 {
  analog_signal = filtered_signal /127;
 }
 else
 {
  analog_signal = -filtered_signal /128;
 }
 if (analog_signal > 0,25)
 {
  if (TF0 == 1)
  {
   if (delay_count == 0)
   {
    BUZZ = 0;
    TR0 = 0;
    TF0 = 0;
    TH0 = 0x3C;
    TL0 = 0xAF;
    delay_count = 100;
   }
   else
   {
    TR0 = 0;
    TF0 = 0;
    TH0 = 0x3C;
    TL0 = 0xAF;
    delay_count --;
    TR0 = 1;
   }
  }
```

```
            else
            {
              BUZZ=1;
              TR0=1;
            }
          }
          else
          {
            if (TR0==1)
            {
              if (TF0==1)
              {
                if (delay_count==0)
                {
                  BUZZ=0;
                  TR0=0;
                  TF0=0;
                  TH0= 0x3C;
                  TL0 = 0xAF;
                  delay_count = 100;
                }
                else
                {
                  TR0=0;
                  TF0=0;
                  TH0= 0x3C;
                  TL0= 0xAF;
                  delay_count--;
                  TR0=1;
                }
              }
            }
```

```
|  |  |  else
|  |  |  {
|  |  |  |  Buzz = 0;
|  |  |  }
|  |  |
|  |  }
|  }
}

int IIR_BPF (float *filter, int input)
{
|  float filterout = 0;
|  filter[2]= filter[1];
|  filter[1]= filter[0];
|  filter[0]= input + 1,912443* filter[1] - 0,986996*filter[2];
|  filterout = 0,006502*filter[0] - 0,006502* filter[2];
|  return (int)(filterout);
}
```