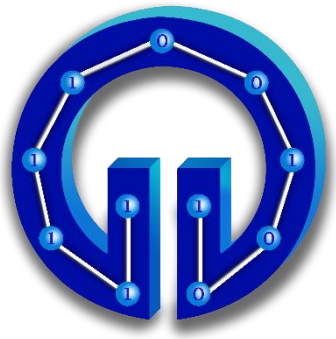
KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ



SAYISAL İŞARET İŞLEME

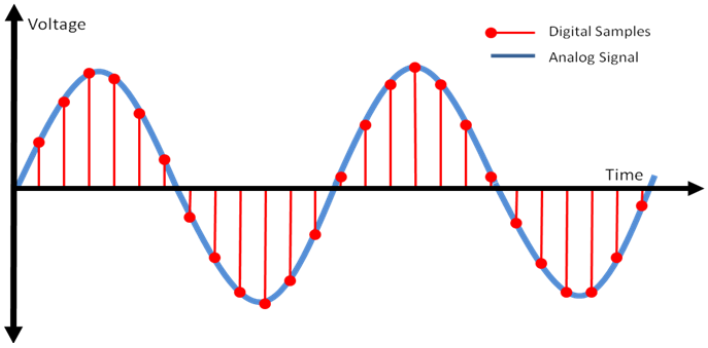
DÖNEM PROJESİ

EMRE AKTÜRK
SERHAN ZEYBEK
MUSTAFA KILINÇ

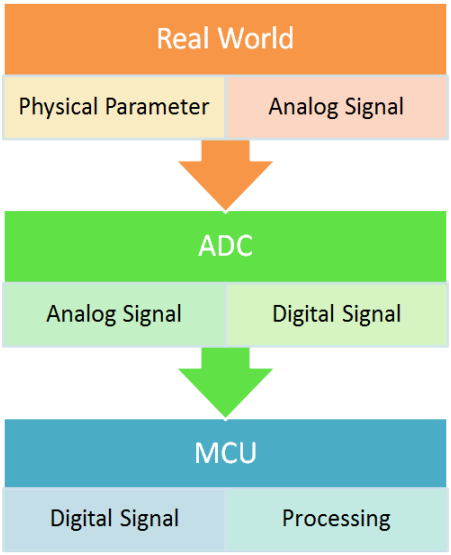
2018-2019 BAHAR DÖNEMİ

ANALOG DIGITAL CONVERTER

Çevremizdeki tüm olaylar analog veriler olup genellikle sinüsoidal sinyallerden oluşur. Bilgisayar ortamında bu sinyalleri ölçüp, işleyip bu veriler üzerinden işlemler yapabilmek için analog digital çeviricilere ihiyacımız vardır. ADC, analog sinyali bizim dijital sistemimizin algılayabileceği dijital veri haline çeviren elektronik devreye denir.



ARDUINO UNO ADC



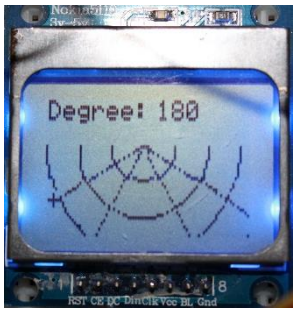
Atmega16/32 mikrodenetleyiciler 8 kanal 10 bit çözünürlüğü olan analog-digital çeviricileri sahiptir. 5 Volt referans değeri kullağımızda 0 ile 5 volt arası 2^10 = 1024 adıma ayrılmıştır. Yani 2.5 Volt yaklaşık 512 değerine karşılık gelmektedir. Projede sensörler 3.3v ile beslenmiştir. Bunun nedeni 3.3 Volt ile örneklenen sinyal üzerinde daha az gürültü olmasıdır.

MAX4466 MODULE

Projede arduino uno kullanılıp iki adet dahili yükselteçe sahip MAX4466 kapasitif mikrofon modülleri kullanılmıştır. 20Hz ile 20kHz frekans aralığında çalışmakta olup üzerinde bulunan op-amp ve potansiyometre ile giriş sinyalini 25-125 kat yükseltme imkanı sağlar.

NOKIA 5110 LCD

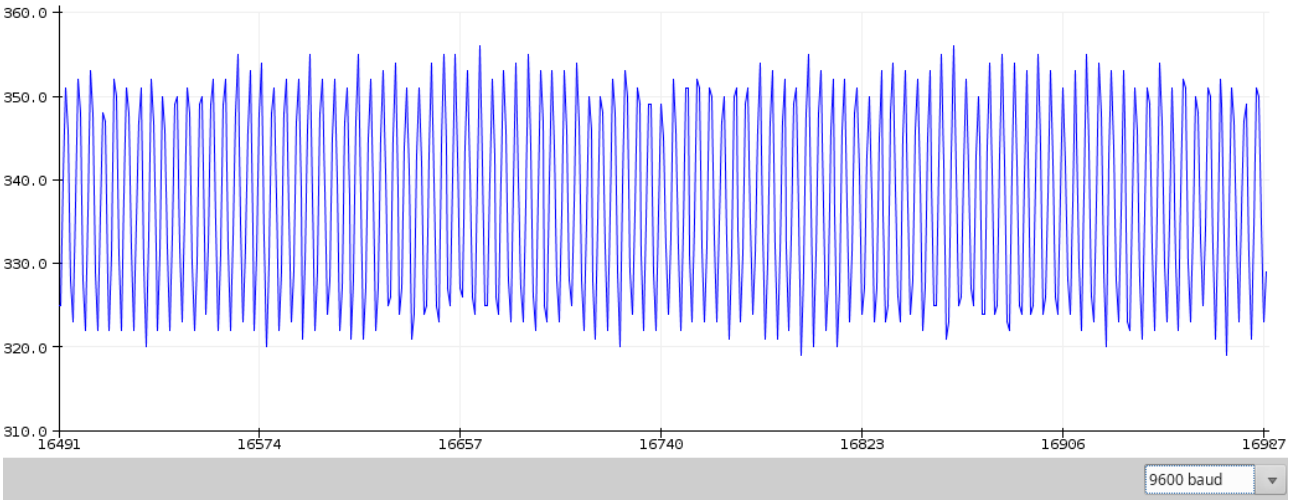
Nokia LCD modülü, Philps'in PCD8544 sürücüsünü kullanmaktadır (Nokia 3310 sürücüsü). 84x48 piksellik grafik ekranlarda sıklıkla kullanılan bir sürücüdür. Bu sebeple bir çok örnek uygulama ve kütüphane bulunmaktadır. Ekran Bitmp resim çizme işlemi için kullanılan program **LCD Assistant**'dır.



ANALOG INPUT ANALİZİ

```
#define MicSamples (1024*2)
#define MicPin A0

// measure basic properties of the input signal
// determine if analog or digital, determine range and average.
void MeasureAnalog()
{
    long signalAvg = 0, signalMax = 0, signalMin = 1024, t0 = millis();
    for (int i = 0; i < MicSamples; i++)
    {
        int k = analogRead(MicPin);
        signalMin = min(signalMin, k);
        signalMax = max(signalMax, k);
        signalAvg += k;
    }
    signalAvg /= MicSamples;

    // print
    Serial.print("Time: " + String(millis() - t0));
    Serial.print(" Min: " + String(signalMin));
    Serial.print(" Max: " + String(signalMax));
    Serial.print(" Avg: " + String(signalAvg));
    Serial.print(" Span: " + String(signalMax - signalMin));
    Serial.print(", " + String(signalMax - signalAvg));
    Serial.print(", " + String(signalAvg - signalMin));
    Serial.println("");
}
```

1KHz Sinus dalgası test edildiğinde;

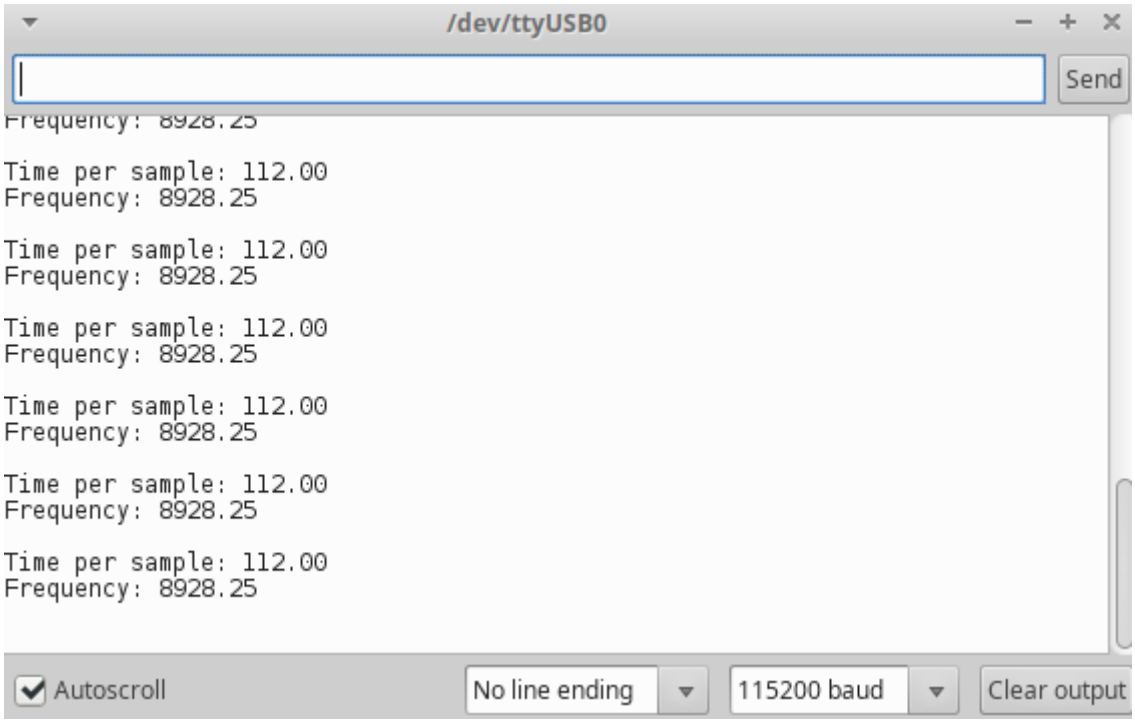


ARDUINO ADC ÖRNEKLEME HIZI

Arduinodaki hazır fonksiyon olan **analogRead()**, hız gerektiren örnekler için iyi bir tercih olmayabilir. Örnek olarak 1000 kere analogRead() fonksiyonun çağırdığımızda;

```
void setup()
{
  Serial.begin(115200);
  pinMode(A0, INPUT);
}

void loop()
{
  long t0, t;

  t0 = micros();
  for(int i=0; i<1000; i++) {
    analogRead(A0);
  }
  t = micros()-t0;  // calculate elapsed time

  Serial.print("Time per sample: ");
  Serial.println((float)t/1000);
  Serial.print("Frequency: ");
  Serial.println((float)1000*1000000/t);
  Serial.println();
  delay(2000);
}
```

Her bir örnek 112 microsecond bu da 8928 Hz örnekleme frekansına eşittir. Örnekleme frekansını port manipulation yöntemini kullanarak yani doğrudan register erişimi yaparak istediğimiz hızı ayarlayabiliriz.



ATMEGA16/32 ADC REGISTER

ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ADCL and ADCH – ADC Data Registers

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1

ATMEGA16/32 ADC PRESCALE TABLOSU

Teorik olarak berlirlenmiş adc örnekleme hızları tabloda verilmiştir.

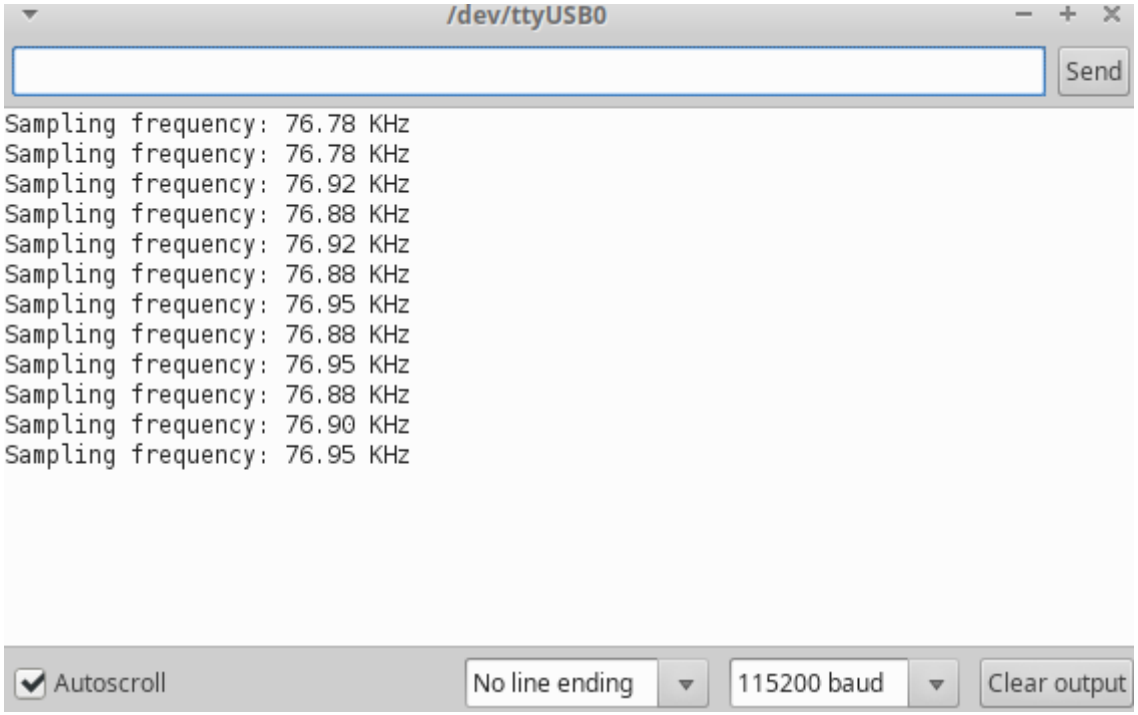
Arduino UNO/ Mega ADC speed= 16MHz clock								
Prescale	ADPS2,1,0	Clock(MHz)	ADC cycle	fs= Sample rate(kHz)	1/fs = ms	Fmax=fs/2 = BW(kHz)	Bits Resolution	
2	0 0 1	8	13	615	0.00162602	307	?	
4	0 1 0	4	13	307	0.00325733	153	8 bit	
8	0 1 1	2	13	153	0.00653595	76.8	8 bit	
16	1 0 0	1	13	76.8	0.01302083	38.4	9 bit	
32	1 0 1	0.5	13	38.4	0.02604167	19.2	10 bit	
64	1 1 0	0.25	13	19.2	0.05208333	9.6	10 bit	
128	1 1 1	0.125	13	9.6	0.10416667	4.8	10 bit	(default)

ARDUINO INTERRUPTS

Arduinonun **analogRead()** fonksiyonu yerine ADC Free Running modunda kullanmak hız gerektiren işlemlerde avantajlı bir yöntemdir. Birinci avantajı cpu zamanını boş yere harcamamış olur. Analog veri geldiğinde ISR(ADC_vect) hemen sensor ile ilgilenir. İkincisi daha hızlı ve hassas şekilde ayrık zamanda örnekleme yapmış oluruz.

```
int numSamples=0;
long t, t0;

void setup()
{
  Serial.begin(115200);

  ADCSRA = 0;             // clear ADCSRA register
  ADCSRB = 0;             // clear ADCSRB register
  ADMUX |= 0x00;          // set A0 analog input pin
  ADMUX |= (1 << REFS0);  // set reference voltage

  // sampling rate is [ADC clock] / [prescaler] / [conversion clock cycles]
  // for Arduino Uno ADC clock is 16 MHz and a conversion takes 13 clock cycles
  //ADCSRA |= (1 << ADPS2) | (1 << ADPS0);    // 32 prescaler for 38.5 KHz
  ADCSRA |= (1 << ADPS2);                     // 16 prescaler for 76.9 KHz
  //ADCSRA |= (1 << ADPS1) | (1 << ADPS0);    // 8 prescaler for 153.8 KHz

  ADCSRA |= (1 << ADATE); // enable auto trigger
  ADCSRA |= (1 << ADIE);  // enable interrupts when measurement complete
  ADCSRA |= (1 << ADEN);  // enable ADC
  ADCSRA |= (1 << ADSC);  // start ADC measurements
}
```

```
ISR(ADC_vect)
{
  int data = ADC;
  numSamples++;
}

void loop()
{
  if (numSamples>=1000)
  {
    t = micros()-t0;  // calculate elapsed time

    Serial.print("Sampling frequency: ");
    Serial.print((float)1000000/t);
    Serial.println(" KHz");
    delay(2000);

    // restart
    t0 = micros();
    numSamples=0;
  }
}
```



BİRDEN FAZLA ANALOG GİRİŞ SEÇİMİ

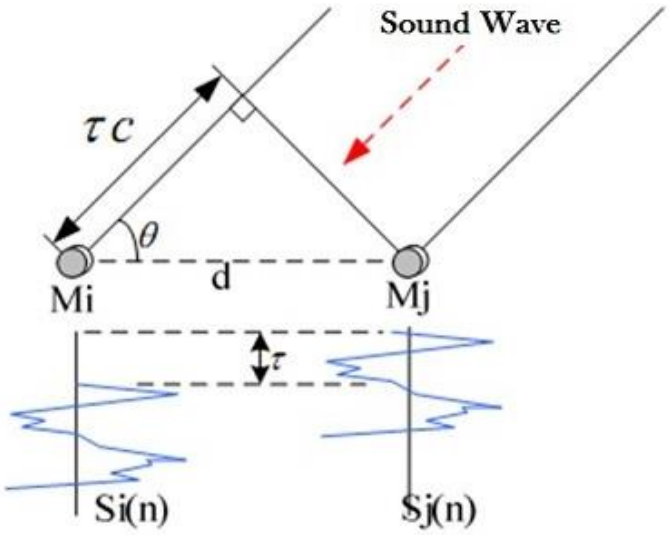
Arduino Uno 8 analog girişe sahibtir ama ADC aynı anda sadece bir analog girişi sayısallaştırabilir. Bu yüzden analog veriyi sayısallaştırmadan önce analog girişi ve referans voltajı girilmelidir. İki farklı şekilde analog pin seçimi aşağıda verilmiştir.

```
void loop() {
  if (analogPin == 3) {
    analogPin = 5;}
  else {
    analogPin = 3;}
  int aval = analogRead(analogPin);
```

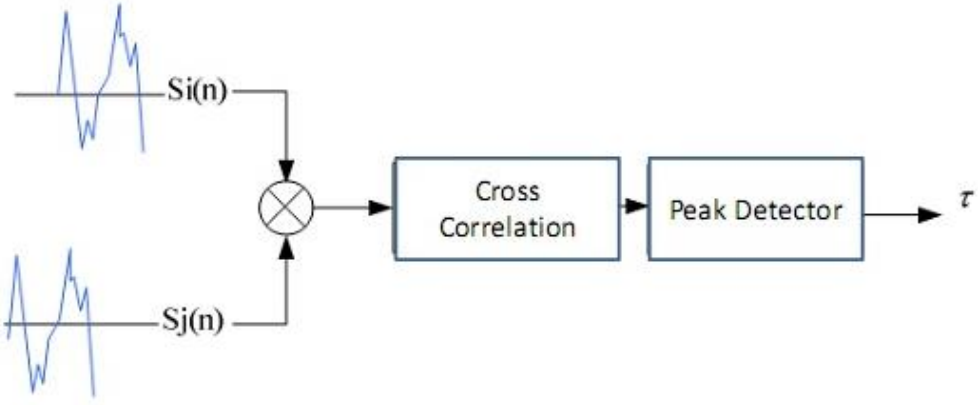
```
ISR(ADC_vect) {
  if (ADMUX == 0xC3) {
    ADMUX = 0xC5;}
  else {
    ADMUX = 0xC3;}
  int aval = ADCL;
```

TDOA YÖNTEMİ İLE KONUM BULMA

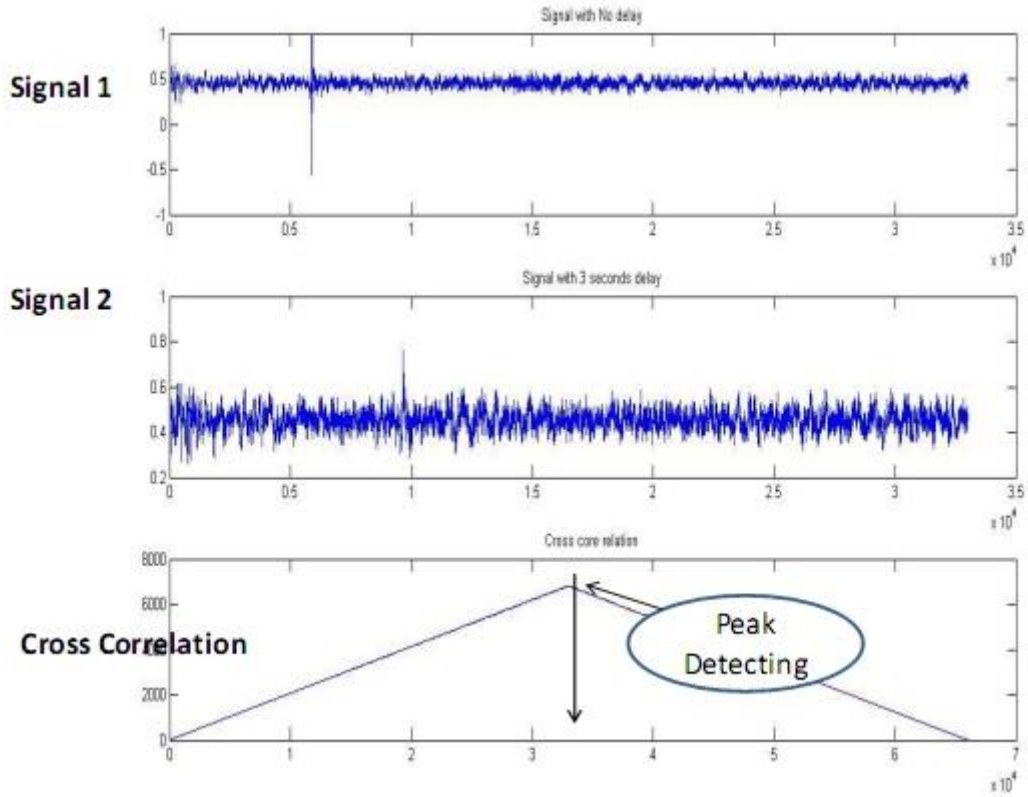
TDOA(Time Difference of Arrival), göndericilerden alıcıya gelen sinyallerin gönderici de oluşan zaman gecikmesine varış zaman farkı denir. Zamanda(Time Domain) konum bulmaya çalışıyorsak iki mikrofon kullanıldığında kaynaktan gelen ses dalgası ADC ile sayısallaştırıldıktan sonra belirlenen örnek sayısı kadar örneklenir.



Gelen sinyaller arasındaki zaman geçikmesi Cross Correlation yapılarak peak değerinin zaman eksenindeki karşılığıdır.



Örnek olarak iki sinyal arasında Cross Correlation yapılarak bulunmuş zaman farkı;



En son olarak basit geometrik işlemler kullanılarak sesin yönünü buluyoruz.

$$Cos\,\theta = \frac{\tau \times C}{d}$$

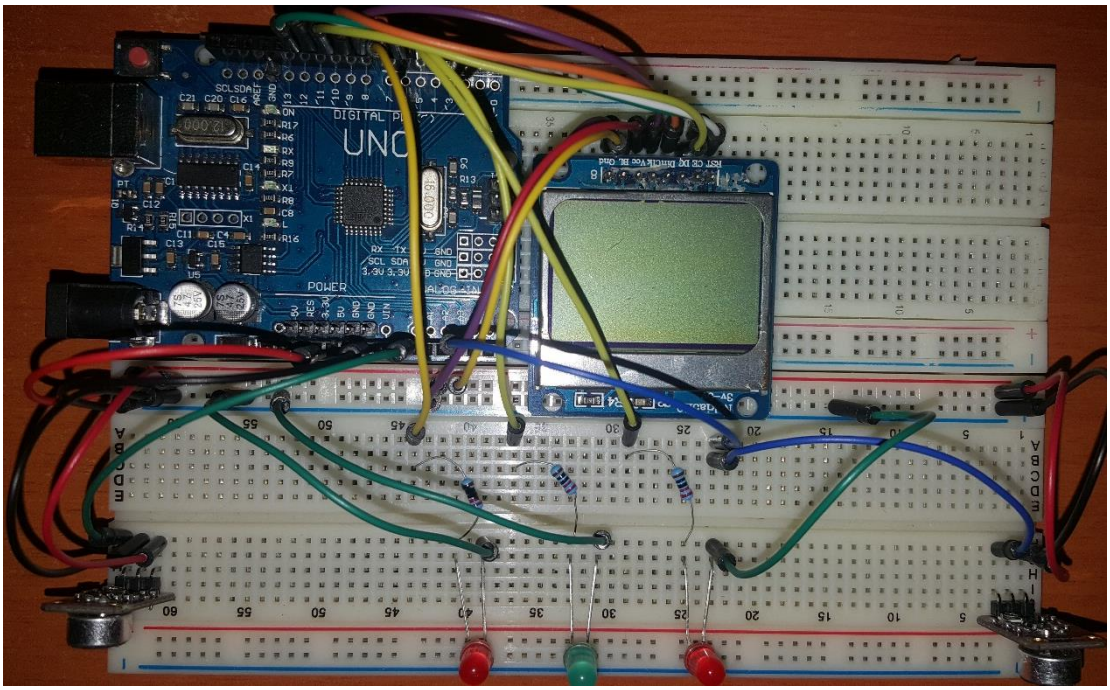
$$\theta = Cos^{-1}\left(\frac{\tau \times C}{d}\right)$$

C = Sesin Hızı(340 m/s)
T = Zaman Farkı
d = Mikrofon Mesafesi

```
#include <math.h>
#include <LCD5110_Basic.h>
#define SAMPLE     10000
#define PTP        30

extern uint8_t SmallFont[];
extern uint8_t icons[];
static char text[] = "Degree: ";
LCD5110 myGLCD(8,9,10,11,12);

static double data = 0;
static int channel = 0;

//double mic1[SAMPLE], mic2[SAMPLE];
long t1 , t2;
long t = 0;
unsigned int count1 = 0, count2 = 0;
static int signalAvg[2] = {0, 0}, signalMax[2] = {0, 0}, signalMin[2] = {679, 679},
peak[2] = {0, 0};

void setup() {

  Serial.begin(115200);
  myGLCD.InitLCD();
  myGLCD.setFont(SmallFont);
  DDRD |= B11100000;

  cli();
  ADCSRA = 0;             // clear ADCSRA register
  ADCSRB = 0;             // clear ADCSRB register
  ADMUX |= 0x00;          // set A0-A1 analog input pin

  ADMUX |= (1 << REFS0);  // set reference voltage
  //ADMUX |= (1 << ADLAR);  // left align ADC value to 8 bits from ADCH register

  // sampling rate is [ADC clock] / [prescaler] / [conversion clock cycles]
  // for Arduino Uno ADC clock is 16 MHz and a conversion takes 13 clock cycles

  //ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);    // 128 prescaler for 9.6
KHz
  ADCSRA |= (1 << ADPS2) | (1 << ADPS0);                 // 32 prescaler for 38.5 KHz
  //ADCSRA |= (1 << ADPS2);                                 // 16 prescaler for 76.9 KHz
  //ADCSRA |= (1 << ADPS1) | (1 << ADPS0);                 // 8 prescaler for 153.8 KHz

  //ADCSRA |= (1 << ADATE); // enable auto trigger
  ADCSRA |= (1 << ADIE);  // enable interrupts when measurement complete
  ADCSRA |= (1 << ADEN);  // enable ADC
  sei();

  ADCSRA |= (1 << ADSC);  // start ADC measurements
  t = micros();
}

ISR(ADC_vect)
{
    data = (int)ADC;
    //Serial.println(data);

    switch(ADMUX)
    {
      case 0x40:
        if(count1 < SAMPLE)
```

```
        {
          count1++;
          //mic1[count1++] = data;
          signalMin[0] = min(signalMin[0], data);
          signalMax[0] = max(signalMax[0], data);
          signalAvg[0] += data;
          if((peak[0] < signalMax[0] - signalMin[0]) && (signalMax[0] - signalMin[0] >
PTP))
          {
            peak[0] = signalMax[0] - signalMin[0];
            t1 = micros() - t;
          }
        }
        ADMUX = 0x43;
        break;
      case 0x43:
        if(count2 < SAMPLE)
        {
          count2++;
          //mic2[count2++] = data;
          signalMin[1] = min(signalMin[1], data);
          signalMax[1] = max(signalMax[1], data);
          signalAvg[1] += data;
          if((peak[1] < signalMax[1] - signalMin[1]) && (signalMax[1] - signalMin[1] >
PTP))
          {
            peak[1] = signalMax[1] - signalMin[1];
            t2 = micros() - t;
          }
        }
        ADMUX = 0x40;
        break;
      default:
        // Default Code
        break;
    }
    ADCSRA |= (1 << ADSC);
}
void loop()
{
  if(count1 >= SAMPLE && count2 >= SAMPLE)
  {
    /*
    for(int i=0; i<SAMPLE; i++)
    {
      Serial.println(mic1[i]);
      Serial.println(mic2[i]);
    }
    */

    int x = 0;
    PORTD &= B00000011;

    if(t1 == 0 && t2 == 0)
    {
      PORTD |= B01000000;
    }
    else if((t1 < t2 || (t2 == 0)) && t1 > 0)
    {
      x = -1;
      PORTD |= B10000000;
    }
```

```
    else
    {
      x = 1;
      PORTD |= B00100000;
    }

    double degree = acos(x) * (180.0 / PI);
    draw(degree);

    peak[0] = 0, signalAvg[0] = 0, signalMax[0] = 0, signalMin[0] = 679;
    peak[1] = 0, signalAvg[1] = 0, signalMax[1] = 0, signalMin[1] = 679;

    count1 = count2 = 0;
    t1 = t2 = 0;
    t = micros();
  }
}

void draw(int degree)
{
  myGLCD.clrScr();
  myGLCD.print(text, LEFT , 0);
  myGLCD.printNumI(degree, 45, 0);
  myGLCD.drawBitmap(0, 16, icons, 84, 48);
  int temp = map(degree, 180, 0, 0, 75);
  myGLCD.print("+", temp , 32);
  myGLCD.invert(false);
  delay(100);
}
```



KAYNAKLAR

[1].blog.yavilevich.com/2016/08/arduino-sound-level-meter-and-spectrum-analyzer
[2].yaab-arduino.blogspot.com/2015/02/fast-sampling-from-analog-input.html
[3].meettechniek.info/embedded/arduino-analog.html
[4].soundlocalize.blogspot.com
[5].researchgate.net/publication/221911943_Sound_Source_Localization_Method_Using_Region_Selection
[6].people.engr.ncsu.edu/kay/msf/sound.htm
[7]. arduino.stackexchange.com/questions/52487/sound-localization-using-two-mics-arduino-uno-on-matlab?rq=1
[8].aktifbeyin.wordpress.com/2016/05/21/time-difference-of-arrival-tdoa/
[9].en.radzio.dxp.pl/bitmap_converter/