Güneş Takip Sistemi

Sefa SAYRACI

İçindekiler Tablosu

**Bölüm başlığını yazın (düzey 1)1**

Bölüm başlığını yazın (düzey 2)2

Bölüm başlığını yazın (düzey 3)3

**Bölüm başlığını yazın (düzey 1)4**

Bölüm başlığını yazın (düzey 2)5

Bölüm başlığını yazın (düzey 3)6

**Güneş Takip Sistemi Sensörler ve Yazılım Alt Sistemi**

1. **Haberleşme Protokolleri**

**UART:** UART açılımı Universal Asynchronous Receiver Transmitter olarak Asenkron haberleşme olduğu anlaşılmaktadır. Seri iletişim için evrensel asenkron alıcı/vericiler (UART'lar) kullanılır. Seri iletişim, iki cihaz arasında her seferinde bir bit veri göndermenin bir yoludur. UART'lar genellikle mikrodenetleyicileri bilgisayarlar, sensörler ve aktüatörler gibi diğer cihazlara bağlamak için kullanılır.

**SPI:** SPI açılımı Serial Peripheral Interface olarak bilinmektedir. Seri çevresel arayüz (SPI), yüksek hızlı bir seri iletişim protokolüdür. SPI genellikle mikrodenetleyicileri flash bellek ve SD kartlar gibi bellek aygıtlarına bağlamak için kullanılır.

**I^2C:** I2C açılımı Inter-Integrated Circuit, Entegre devreler arası (I2C) düşük hızlı bir seri iletişim protokolüdür. I2C genellikle mikrodenetleyicileri sensörlere ve aktüatörlere bağlamak için kullanılır.

**RS232:** Telekomünikasyonda RS-232, DTE (veri terminal ekipmanı) ile DCE (veri taşıma ekipmanı) arasındaki seri ikili tek sonlu veri iletimi ve sinyalleme için kullanılan seri iletişim standardının genel adıdır. Daha çok bilgisayardaki seri portlarda kullanılır. RS-232 (ANSI / EIA-232 standardı), IBM-PC ve uyumlu bilgisayarlarda seri bağlantı standardıdır. Fare, yazıcı veya modem bağlamak gibi birçok amaç için kullanılabilir ve endüstriyel enstrümanları da bağlayabilir. Sürüş ve kablolamayı geliştirmek için kullanılır. Uygulamada, RS-232’nin iletim uzunluğu veya hızı genellikle standart değeri aşar. RS-232, PC seri portu ile cihaz arasındaki noktadan noktaya iletişim ile sınırlıdır. RS-232 ile seri iletişim yaklaşık 15 metre kadardır.

**RS422:** Bu standart, RS-232C’deki kısa iletim mesafesi ve yavaş iletim hızı gibi sorunları düzeltir. “EIA-422A” olarak da adlandırılır. Sinyal hatlarının amacı ve zamanlaması tanımlanır, ancak konektörler tanımlanmaz. öncelikle D-sub 25-pin ve D-sub 9-pin konnektörleri benimser. RS-422 diferansiyel sinyaller kullanır ve RS-232 dengesiz toprak referanslı sinyaller kullanır. Diferansiyel iletim, sinyal göndermek ve almak için iki kablo kullanır RS-232 ile karşılaştırıldığında, gürültüye daha iyi direnebilir ve daha uzun iletim mesafesine sahip olabilir. Daha iyi gürültü bağışıklığı ve endüstriyel ortamlarda daha uzun iletim mesafeleri büyük bir artıdır.

**RS485:** RS-485 (EIA-485 standardı) RS-422’nin bir gelişmesidir, çünkü cihaz sayısını 10’dan 32’ye çıkarır ve yeterli sağlamak için maksimum cihaz sayısındaki elektriksel karakteristikleri tanımlar. Birden fazla cihazın yeteneği sayesinde, bir cihaz ağı kurmak için tek bir RS-422 portu kullanabilirsiniz. Mükemmel gürültü önleme ve çoklu cihaz özellikleri endüstriyel uygulamalarda, diğer veri toplama kontrol cihazlarında, HMI’larda veya diğer işlemlerde PC’lere dağıtılmış bir cihaz ağı kurarken, RS-485 seri bağlantıdır.

RS-485 dezavantajı , programlamak daha zor olmasıdır, çünkü veri göndermek ve almak için aynı 2 kabloyu kullanır. Ve herhangi bir ağda, yalnızca bir düğüm veri iletebilir, diğer düğümler yalnızca o anda alabilir. On avantajları tarafında, herhangi bir sorun ile uzun mesafe iletişimi destekler. Ayrıca, arayüz devresinin zarar görmesini zorlaştıran RS-232’den daha düşük arayüz sinyal seviyeleri kullanır.

RS-485, RS-422’nin bir üst kümesidir, bu nedenle tüm RS-422 cihazları RS-485 tarafından kontrol edilebilir. RS-485, 1200 metre telin üzerindeki seri trafik için kullanılabilir.

**CAN Haberleşme:** [**https://rmc.com.tr/can-bus-controller-area-network-nedir-ve-diger-arac-bus-veri-yolu-aglari-ile-nasil-karsilastirilir/**](https://rmc.com.tr/can-bus-controller-area-network-nedir-ve-diger-arac-bus-veri-yolu-aglari-ile-nasil-karsilastirilir/)

**RS232 ve RS485 Arasındaki Farklar:**

|  |  |  |
| --- | --- | --- |
|  | **RS232** | **RS485** |
| **Hat Yapısı** | Tek katlı | Diferansiyel |
| **Aygıt Sayısı** | 1 alıcı, 1 gönderici | 32 alıcı, 32 gönderici |
| **Çalışma Şekli** | Simplex ya da full duplex | Simplex ya da half duplex |
| **Max. Kablo Uzunluğu** | 15 m | 1200 m |
| **Max. Veri Hızı** | 20 Kbits/s | 10 Kbits/s |
| **Sinyalizasyon** | Dengesiz | Dengeli |
| **Tipi Logic Seviyeleri** | +-5 ~ +-15V | +-1.5 ~ +-6V |
| **Min. Alıcı Giriş Empedansı** | 3 ~ 7 K-ohm | 12 K-ohm |
| **Alıcı Duyarlılığı** | +-3V | +-200mV |

**Asenkron ve Senkron Haberleşme Arasındaki Farklar:**

|  |  |
| --- | --- |
| **Asenkron** | **Senkron** |
| Alıcı verici arasında ortak bir saat sinyali yoktur. | İletişim paylaşılan bir saat ile yapılır. |
| Bir sefer 1 byte gönderir. | Veriyi bloklar halinde gönderir. |
| Senkrona göre yavaştır. | Asenkrona göre hızlıdır. |
| Start stop biti nedeniyle ek yük. | Daha az ek yük. |
| Uzak mesafeler. | Asenkrona göre daha kısa mesafeler. |
| Veri senkronizasyonu için start stop biti. | Veri senkronizasyonu için ortak saat kullanımı. |
| RS232, RS485 | I2C, SPI |

1. **Elektronik Alt Sistem Ürün Seçimleri**
2. **Mikroişlemci Seçimi**

**Proje Başında Seçilen Mikroişlemci:**

Esp32 Wroom 32U üzerinden geliştirme yapılacaktır.

**Esp32 Wroom 32U:**

****

**Resim 1.** Esp32 Mikroişlemci Geliştirme Kartı

1. **İvmeölçer + Dönmeölçer + Pusula**

Gyro sensörü olarak 6DOF olacak şekilde ivmeölçer ve dönme ölçer verilerini alacağım. Buna ek olarak Pusula sensörü kullanacağım. 9DOF olarak sensörü eş zamanlı olarak kullanabilirim ancak. Elimde bulunmamasından dolayı bu şekilde bir yol seçtim.

**6 DOF Seçilen Sensör MPU6050:**

****

**Resim 1.1** MPU6050 6DOF Sensör

**HMC5883L Pusula Sensör:**

****

**Resim 1.2** HMC5883L Sensör

1. **Yükseklik Ölçer (Basınç Sensörü)**

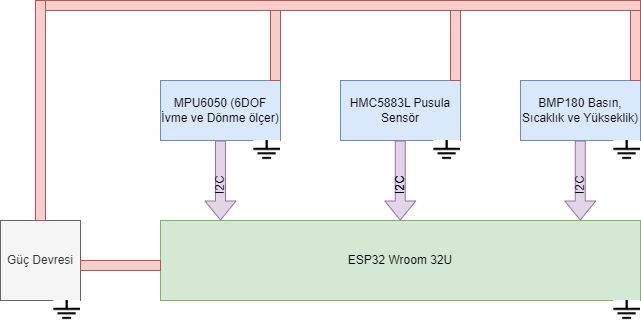
Yükseklik verisi basıç sensöründen eş zamanlı olarak alınacaktır. Bu sensör üzerinden basınç, sıcaklık ve yükseklik verisi gelecektir.

**BMP180:**



**Resim 1.3** BMP180 Sensör

1. **Elektrikseel Bağlantı Şeması && Block Diagram**

****

1. **Kullanılan Haberleşme Protokolleri**

Kullanılan sensörlerin hepsinde I2C haberleşemesi sağlanmaktadır. Düşük hızda senkron haberleşme yapılmaktadır. Bilgisayar üzerinden serial port üzerinden alınacak veriler string tür şeklinde UART (Asenkron Haberleşme) protokolü ile haberleşmektedir. Bu COM soketi üzerinden C# üzerinde hazırlanacak olan arayüz üzerinden anlık ayalanacak süre ile buffer ı sürekli temizlenmesiyle sağlanacaktır.

1. **Sensör Kodları**

**Mpu6050 Kodu:**

[**https://randomnerdtutorials.com/esp32-mpu-6050-accelerometer-gyroscope-arduino/**](https://randomnerdtutorials.com/esp32-mpu-6050-accelerometer-gyroscope-arduino/)

#include <Adafruit\_MPU6050.h>

#include <Adafruit\_Sensor.h>

#include <Wire.h>

Adafruit\_MPU6050 mpu;

*void* *setup*(*void*) {

  Serial.*begin*(115200);

  while (!Serial)

*delay*(10); *// will pause Zero, Leonardo, etc until serial console opens*

  Serial.*println*("Adafruit MPU6050 test!");

*// Try to initialize!*

  if (!mpu.*begin*()) {

    Serial.*println*("Failed to find MPU6050 chip");

    while (1) {

*delay*(10);

    }

  }

  Serial.*println*("MPU6050 Found!");

  mpu.*setAccelerometerRange*(MPU6050\_RANGE\_8\_G);

  Serial.*print*("Accelerometer range set to: ");

  switch (mpu.*getAccelerometerRange*()) {

  case MPU6050\_RANGE\_2\_G:

    Serial.*println*("+-2G");

    break;

  case MPU6050\_RANGE\_4\_G:

    Serial.*println*("+-4G");

    break;

  case MPU6050\_RANGE\_8\_G:

    Serial.*println*("+-8G");

    break;

  case MPU6050\_RANGE\_16\_G:

    Serial.*println*("+-16G");

    break;

  }

  mpu.*setGyroRange*(MPU6050\_RANGE\_500\_DEG);

  Serial.*print*("Gyro range set to: ");

  switch (mpu.*getGyroRange*()) {

  case MPU6050\_RANGE\_250\_DEG:

    Serial.*println*("+- 250 deg/s");

    break;

  case MPU6050\_RANGE\_500\_DEG:

    Serial.*println*("+- 500 deg/s");

    break;

  case MPU6050\_RANGE\_1000\_DEG:

    Serial.*println*("+- 1000 deg/s");

    break;

  case MPU6050\_RANGE\_2000\_DEG:

    Serial.*println*("+- 2000 deg/s");

    break;

  }

  mpu.*setFilterBandwidth*(MPU6050\_BAND\_5\_HZ);

  Serial.*print*("Filter bandwidth set to: ");

  switch (mpu.*getFilterBandwidth*()) {

  case MPU6050\_BAND\_260\_HZ:

    Serial.*println*("260 Hz");

    break;

  case MPU6050\_BAND\_184\_HZ:

    Serial.*println*("184 Hz");

    break;

  case MPU6050\_BAND\_94\_HZ:

    Serial.*println*("94 Hz");

    break;

  case MPU6050\_BAND\_44\_HZ:

    Serial.*println*("44 Hz");

    break;

  case MPU6050\_BAND\_21\_HZ:

    Serial.*println*("21 Hz");

    break;

  case MPU6050\_BAND\_10\_HZ:

    Serial.*println*("10 Hz");

    break;

  case MPU6050\_BAND\_5\_HZ:

    Serial.*println*("5 Hz");

    break;

  }

  Serial.*println*("");

*delay*(100);

}

*void* *loop*() {

*/\* Get new sensor events with the readings \*/*

  sensors\_event\_t a, g, temp;

  mpu.*getEvent*(&a, &g, &temp);

*/\* Print out the values \*/*

  Serial.*print*("Acceleration X: ");

  Serial.*print*(a.acceleration.x);

  Serial.*print*(", Y: ");

  Serial.*print*(a.acceleration.y);

  Serial.*print*(", Z: ");

  Serial.*print*(a.acceleration.z);

  Serial.*println*(" m/s^2");

  Serial.*print*("Rotation X: ");

  Serial.*print*(g.gyro.x);

  Serial.*print*(", Y: ");

  Serial.*print*(g.gyro.y);

  Serial.*print*(", Z: ");

  Serial.*print*(g.gyro.z);

  Serial.*println*(" rad/s");

  Serial.*print*("Temperature: ");

  Serial.*print*(temp.temperature);

  Serial.*println*(" degC");

  Serial.*println*("");

*delay*(500);

}

**HMC5883L Kodu:**

[**http://www.esp32learning.com/code/an-esp32-and-hmc5883l-sensor-example.php**](http://www.esp32learning.com/code/an-esp32-and-hmc5883l-sensor-example.php)

#include <Wire.h>

#include <Adafruit\_Sensor.h>

#include <Adafruit\_HMC5883\_U.h>

*/\* Assign a unique ID to this sensor at the same time \*/*

Adafruit\_HMC5883\_Unified mag = *Adafruit\_HMC5883\_Unified*(12345);

*void* *setup*(*void*)

{

Serial.*begin*(9600);

*/\* Initialise the sensor \*/*

if(!mag.*begin*())

{

*/\* There was a problem detecting the HMC5883 ... check your connections \*/*

Serial.*println*("Ooops, no HMC5883 detected ... Check your wiring!");

while(1);

}

}

*void* *loop*(*void*)

{

*/\* Get a new sensor event \*/*

sensors\_event\_t event;

mag.*getEvent*(&event);

*/\* Display the results (magnetic vector values are in micro-Tesla (uT)) \*/*

Serial.*print*("X: ");

Serial.*print*(event.magnetic.x);

Serial.*print*(" ");

Serial.*print*("Y: ");

Serial.*print*(event.magnetic.y);

Serial.*print*(" ");

Serial.*print*("Z: ");

Serial.*print*(event.magnetic.z);

Serial.*print*(" ");

Serial.*println*("uT");

*// Hold the module so that Z is pointing 'up' and you can measure the heading with x&y*

*// Calculate heading when the magnetometer is level, then correct for signs of axis.*

*float* heading = *atan2*(event.magnetic.y, event.magnetic.x);

*// Once you have your heading, you must then add your 'Declination Angle', which is the 'Error' of the magnetic field in your location.*

*// Find yours here: http://www.magnetic-declination.com/*

*// Mine is: -13\* 2' W, which is ~13 Degrees, or (which we need) 0.22 radians*

*// If you cannot find your Declination, comment out these two lines, your compass will be slightly off.*

*float* declinationAngle = 0.22;

heading += declinationAngle;

*// Correct for when signs are reversed.*

if(heading *<* 0)

heading += 2\*PI;

*// Check for wrap due to addition of declination.*

if(heading *>* 2\*PI)

heading -= 2\*PI;

*// Convert radians to degrees for readability.*

*float* headingDegrees = heading \* 180/M\_PI;

Serial.*print*("Heading (degrees): ");

Serial.*println*(headingDegrees);

*delay*(500);

}

**BMP180 Kodu:**

[**https://www.electronicshub.org/esp32-bmp180-tutorial/**](https://www.electronicshub.org/esp32-bmp180-tutorial/)

#include <Adafruit\_BMP085.h>

Adafruit\_BMP085 bmp;;

*void* *setup*()

{

  Serial.*begin*(115200);

  if (!bmp.*begin*())

  {

    Serial.*println*("BMP180 Sensor not found ! ! !");

    while (1)

    {

    }

  }

}

*void* *loop*()

{

  Serial.*print*("Pressure = ");

  Serial.*print*(bmp.*readPressure*());

  Serial.*print*(" hPa");

  Serial.*print*("  Temp = ");

  Serial.*print*(bmp.*readTemperature*());

  Serial.*println*("ºC");

*delay*(3000);

}

1. **Sistem Kodları**

Sistem kod birleşimi Arduino üzerinden oluşturulmuştur.

Sistem tüm kodları şu şekildedir;

1. **Telemetri Arayüzü**

Tüm sensörlerden gelen veriler eş zamanlı olarak birer saniye (1 Hz) aralıklarla anlık olarak gösterilecektir. Aynı zamanda bu verilerden sıcaklık, basınç ve yükseklik verileri grafik üzerinde gösterilecektir. Bu application üzerinde serial port ayarları olarak baudrate ve Com port seçme özellikleri de mevcuttur.

Geliştirilen bu arayüzde parolalı giriş mevcuttur. Bu parolanın girişi sonrasında arayüz üzerinde serial port ile bağlantı seçildikten sonra veriler eş zamanlı gelecektir. Bu verilerin gelmesi sonucu buffer ın sürekli temizlenerek arkada bir thread çalıştırarak veri yoğunluğu silinir ve arayüz programının donma problemi yaşamasını engeller.