Стабилизатор за екшън камера

Мирослав Николов
Деян Златанов

Съдържание

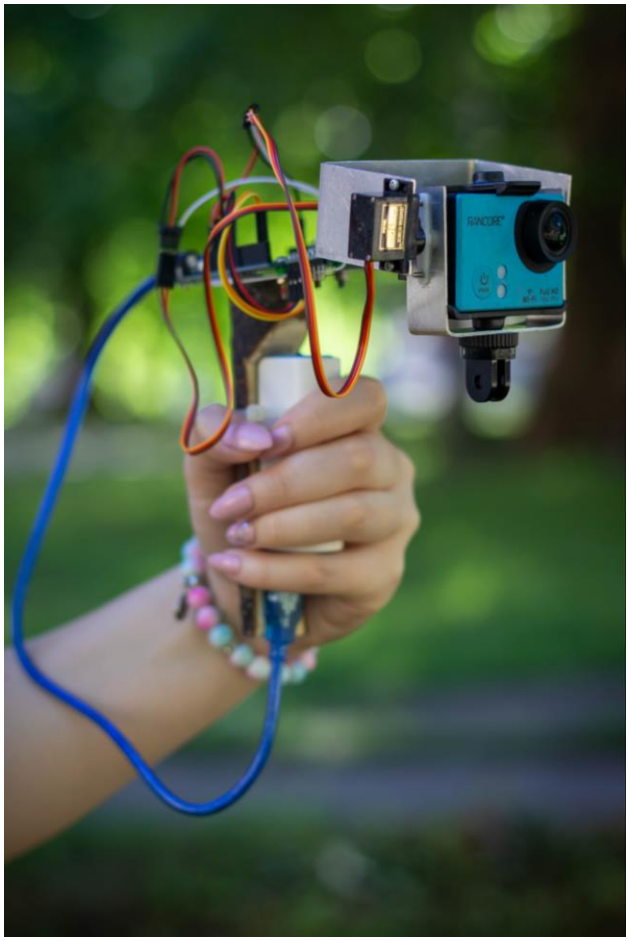
Въведение

Цели

Целта на този проект е създаване на уред с помощта на ардуино микроконтролер, който да стабилизира и държи екшън камера перпендикулярно на земята.

Използвани компоненти

Елемент	Количество
Arduino Uno R3	1
MPU6050 Accelerometer and Gyroscope	1
SG90 Mirocro Servo	2



Описание на проекта

Действие на компонентите

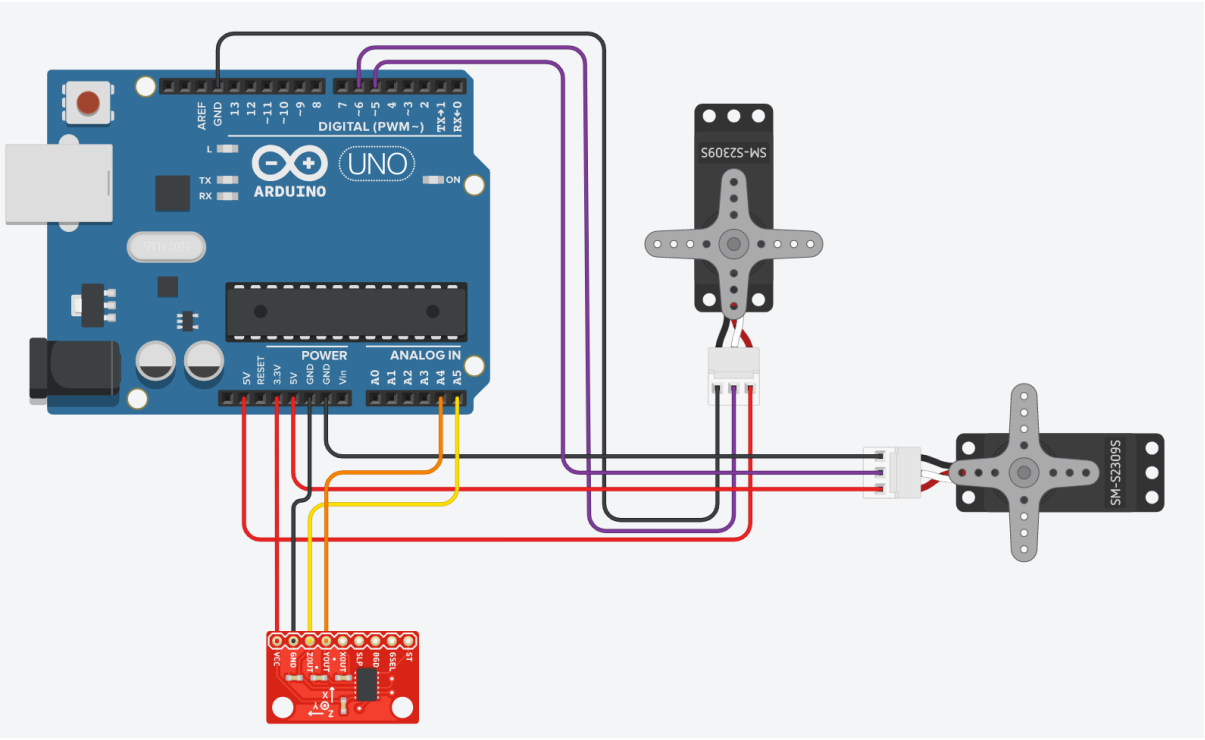
- Ардуино Уно
    - Микроконтролер, който приема, обработва, получената от акселерометъра информация, и изпраща към серво моторите, с колко градуса да се завъртят.
- 3 осов жироскоп и акселерометър MPU6050
    - Неговото предназначение е да следи отклонението при завъртане по 3-те му оси.
- Серво мотори
    - Тяхната задача е да завъртят камерата на определена позиция

Принцип на работа

Един от най-важните компоненти в нашия проект е акселерометърът. Той измерва какво е земното привличане във всяка една от осите и в зависимост от това връща цифра в диапазона от 0 до ~4000.

Двата серво мотора приемат вече обработената, от ардуиното, информация в градуси и съответно се завъртат на точно обратните градуси, за да противодействат на завъртането.

Схема на свързване



Описание на кода

Трите използвани в проекта библиотеки са: Wire, Servo, Smoothed.

Wire е библиотека за комуникация между ардуиното и акселерометъра посредством I2C портокол.

Servo е библиотека за комуникация между ардуиното и серво моторите.

Smoothed е библиотека за осредняване на взетите резултати и премахване на излишни движения от серво моторите.

Серво моторите са дефинирани на 5 и 6 пин.

За работа със сензора MPU6050 е необходимо да се достъпи директно до определен регистър за определено действие.

За по- удобна работа сме дефинирали регистрите за захранването, четенето, големината на четене и различните настройки за чувствителност.

Дефиниран е и адреса на самия сензор за директно достъпване.

```
#include <Wire.h>
#include <Servo.h>
#include <Smoothed.h>

#define SERVO_X                     5
#define SERVO_Y                     6

#define MPU_POWER_REG               0x6B
#define MPU_POWER_CYCLE             0b00000000

#define MPU_ACCEL_CFG_REG           0x1C
#define MPU_ACCEL_CFG_2G            0b00000000
#define MPU_ACCEL_READINGSCALE_2G   16384.0
#define MPU_ACCEL_CFG_4G            0b00001000
#define MPU_ACCEL_READINGSCALE_4G   8192.0
#define MPU_ACCEL_CFG_8G            0b00010000
#define MPU_ACCEL_READINGSCALE_8G   4096.0
#define MPU_ACCEL_READ_REG          0x3B
#define MPU_ACCEL_READ_REG_SIZE     6

#define MPU_I2C_ADDRESS             0b1101000
```

В метода **setup** отваряме порт за серийна комуникация. Прикачаме серво моторите към опрделените пинове. Започваме I2C комнукация. Настройваме какъв метод и по колко резултата да взима за осредняване.

```
void setup() {
  Serial.begin(9600);
  ServoX.attach(SERVO_X);
  ServoY.attach(SERVO_Y);
  Wire.begin();
  SetupMPU();
  MoveServosToZeroPosition();

  ServoPitchSm.begin(SMOOTHED_EXPONENTIAL, 25);
  ServoRollSm.begin(SMOOTHED_EXPONENTIAL, 25);
}
```

В метода MPUReadAccel започваме да четем суровите данни от акселерометъра.

Четем данните за всяка ос, като взимаме първия байт, преместваме го 8 бита и залепяме следващия байт, за да се получи 16 битов integer

```
void MPUReadAccel(){
  Wire.beginTransmission(MPU_I2C_ADDRESS);
  Wire.write(MPU_ACCEL_READ_REG);
  Wire.endTransmission();
  Wire.requestFrom(MPU_I2C_ADDRESS, MPU_ACCEL_READ_REG_SIZE);

  AcX = (long) (Wire.read() << 8 | Wire.read()) / MPU_ACCEL_READINGSCALE_4G;
  AcY = (long)(Wire.read() << 8 | Wire.read()) / MPU_ACCEL_READINGSCALE_4G;
  AcZ = (long) (Wire.read() << 8 | Wire.read()) / MPU_ACCEL_READINGSCALE_4G;
}
```

В метода MPUCalculatePitchAndRoll изчисляваме колко е наклона настрани и напред по осите X и Y.

Ако си представим, че сензора е едно топче вързано на четири места, така че всяко да е перпендикулярно на съседнотото в квадратна рамка и я поставим под 45 гадусов ъгъл, то силата която дърпа топчето надолу ще е поравно разпределена между горните две въженца. По този начин може да намерим гравитационния вектор и понеже въженцата са перпендикулярни може да използваме питагоровата теорема. След като знаем гравитационния вектор можем да приложим обратен синус. Ардуино функцията връща резултата в радиани и затова умножаваме по 57.296, за да получим резултата в градуси.

```
void MPUCalculatePitchAndRoll(){
  TotalVector = sqrt((AcX * AcX)+ (AcY * AcY)+ (AcZ * AcZ));
  Pitch = asin((float)AcY / TotalVector) * 57.296;
  Roll =  asin((float)AcX / TotalVector) * -57.256;

}
```

Метода MoveServo взима градусите които варират от ~-90 до +90 и чрез функцията map те се преобразуват от 0 до 180 градуса.

Всеки резултат се добавя към библиотека за усредняване и после се взима вече усредения резултат и се подава към серво моторите.

```
void MoveServo(){
  ServoPitchSm.add(map(Pitch, -78, 80, 180, 0));
  ServoRollSm.add(map(Roll, -75, 80, 0, 180));

  ServoPitch = ServoPitchSm.get();
  ServoRoll = ServoRollSm.get();

  ServoX.write(ServoRoll);
  ServoY.write(ServoPitch);
}
```

Метода printData показва на сериинния монитор както суровите данни, така и тези, които се подават към серво моторите.

```
void printData(){
  Serial.print(" ServoPitch: ");
  Serial.print(ServoPitch);
  Serial.print(" ServoRoll: ");
  Serial.print(ServoRoll);
  Serial.print(" Accel (g)");
  Serial.print(" X=");
  Serial.print(AcX);
  Serial.print(" Y=");
  Serial.print(AcY);
  Serial.print(" Z=");
  Serial.println(AcZ);
  Serial.print(" Roll:");
  Serial.print(Roll);
  Serial.print(" Pitch:");
  Serial.print(Pitch);
}
```

Заключение

Бъдещи подобрения

Като подобрения предвиждаме подменяне на серво моторите със стъкови мотори за по- гладко движение.

По- добро филтриране на данните.

Заменяне на Arduino Uno R3 с Arduino micro или nano.

Източници

Работа с MPU6050 :

https://youtu.be/fEHF-m1TR6g

Обработка на суровите данни към приложими:

https://youtu.be/j-kE0AMEWy4