(по материалам

https://www.youtube.com/watch?v=Bl0gzVYJ69I

https://abudawud.wordpress.com/2018/06/01/mengenal-sensor-imu-gy25/)

(DATASHEET: http://mkpochtoi.ru/GY25_MANUAL_EN.pdf

https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf)

Датчик IMU GY-25

GY25-это датчик аналогичный датчикам IMU. Как и датчик imu в целом, в датчике есть датчик ускорения и гироскоп. Разница gy25 с датчиками imu в наличии мощного ARM микроконтроллера, который имеет встроенную программу предварительной обработки данных так, что выход этого датчика готов к использованию без необходимости проведения калмановской фильтрации для всех данных по крену, тангажу и рысканью. Параметр рыскание используется для замены датчика компаса.

Руководство по применению.

1. Параметры

• Диапазон измерения: -180 ° 1 °
• разрешение: 0,01 °
• точность: 1
• Частота отклика: 100 Гц (при скорости обмена 115200 бит / с)
• Размеры: 11,5 мм × 15,5 мм

2. Выводы

- **Pin1 VCC** Power + (3v-5v)
- **Pin 2 RX**
- **Pin 3 TX**
- **Pin 4 GND**
- **Pin 5 RST**
- **Pin 6 B0**
- **Pin 7 SCL** I2C
- **Pin 8 SDA** I2C

3. Связь с устройством

Этот датчик использует серийное сообщение (RX/TX) с выбором baudrate 9600 и 115200. Скорость выбирается с помощью перемычки, помеченной как 1 на изображении выше. Если перемычки нет (по умолчанию), то используется скорость 115200, если установлена то скорость обмена 9600. Скорость 115200 может быть использована для получения быстрого обновления данных gy25.

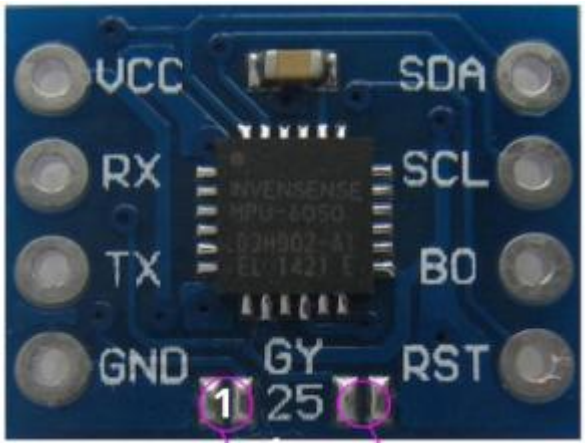
4. Команды

Чтобы иметь возможность конфигурировать и калибровать gy 25, используются следующие команды

1. **0xA5 + 0x51**: режим запроса, отправка данных датчика, когда есть запрос
2. **0xA5 + 0x52**: Автоматический режим, отправка данных датчика в реальном времени в двоичном виде
3. **0xA5 + 0x53**: Автоматический режим, отправка данных датчика в режиме реального времени в виде ASCII
4. **0xA5 + 0x54**: режим коррекции, калибровка шага крена. подождите несколько секунд перед калибровкой
5. **0xA5 + 0x55**: режим коррекции, калибровка компаса. Компас будет установлен в 0 при отправке этой команды

Примечание: (1) Во время автоматической калибровки, когда датчик впервые активен, датчик должен оставаться неподвижным (беззвучным) не менее 3 секунд (держать его в руке не рекомендуется), (2) Модуль gy25 имеет напряжение 5,0 В

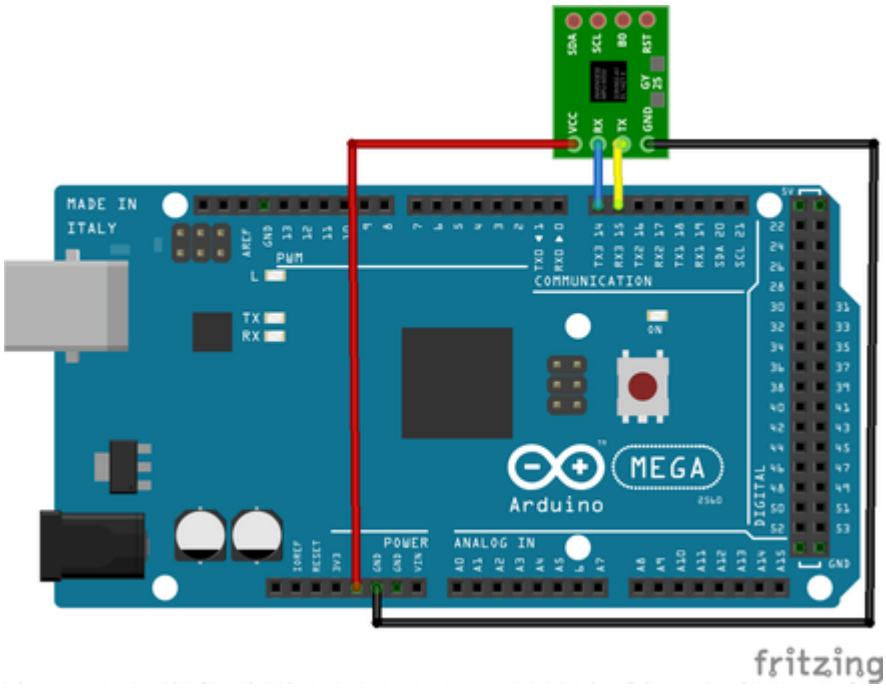
Доступ к данным датчика IMU GY25 с помощью Arduino



GY25 - это датчик IMU с выходными данными в виде крена, шага и рыскания (подробнее). В этой главе описывается процесс доступа к данным GY25 с помощью Arduino Mega. Скорость передачи GY25 составляет 115200 без перемычки 1, на плате GY25.

Схема

Используется следующая схема



1. Подключение

Контакты, подключенные к gy25, представляют собой контакты Serial3 на Arduino Mega, RX3 и TX3. Для тех, кто использует Arduino Uno или Nano, можно использовать Software Serial, чтобы иметь возможность преобразовывать обычные выводы в последовательный

вывод Arduino Software Serial.

2. Скорость обмена.

Скорость передачи данных, используемая в конфигурации настройки Arduino, составляет 115200 (настраивается на скорость передачи GY25.

B. Программирование.

Программирование выполняется с использованием IDE Arduino с платой Arduino Mega 2560.

1. Конфигурирование

Перед тем, как получить доступ к данным датчика GY25, его необходимо предварительно настроить, который включает

- Запуск последовательной связи с 115200 бит / с для Serial Monitor.
- инициализация Serial3 с 115200 бит / с для связи Arduino с GY25
- Калибровка (крен, тангаж и рыскание)
- Калибровка курса
- Конфигурация типа данных (выход ASCII в реальном времени)

a. Serial Monitor

Чтобы начать последовательную связь с ПК / ноутбуком или известным как Serial Monitor, который предназначен для мониторинга данных датчика от GY25, используются следующие команды

```
1void setup(){
2  Serial.begin(115200); // Serial monitor
3}
4
5void loop(){
6}
```

b. Serial3 GY25

Чтобы иметь возможность общаться с GY25, используются следующие команды

```
1void setup(){
2 Serial.begin(115200); // Serial monitor
3 Serial3.begin(115200); // Serial GY25
```

```
}

void loop(){

}
```

c. Калибровка наклона (крен, наклон и рыскание)

Чтобы выполнить калибровку, отправьте команду калибровки со следующим кодом, отметив, что перед калибровкой должна быть пауза (датчик стабилен).

Эта калибровка направлена на регулировку положения датчика в зависимости от наклона датчика.

```
void setup(){
  Serial.begin(115200); // Serial monitor
  Serial3.begin(115200); // Serial GY25
  delay(3000); // Jeda 3 detik

  // Kalibrasi Tilt
  Serial3.write(0xA5);
  Serial3.write(0x54);
 }

void loop(){
 }
```

d. Калибровка курса

Целью калибровки курса является сброс курса на датчике GY25 до 0 при отправке команды сброса курса.

Целью калибровки головки является сброс компаса на датчике GY 25 до 0 при отправке команды сброса курса.

```
void setup(){
  Serial.begin(115200); // Serial monitor
  Serial3.begin(115200); // Serial GY25
```

```
  delay(3000); // Jeda 3 detik

  // Kalibrasi Tilt
  Serial3.write(0xA5);
  Serial3.write(0x54);

  delay(1000); // Jeda sebelum kalibrasi heading

  // Kalibrasi Heading
  Serial3.write(0xA5);
  Serial3.write(0x55);
}

void loop(){
}
```

e. Настройка вывода ASCII

Для облегчения считывания данных датчика с помощью последовательного монитора, вывод данных датчика может быть установлен на ASCII следующим образом

```
void setup(){
  Serial.begin(115200); // Serial monitor
  Serial3.begin(115200); // Serial GY25
  delay(3000); // Jeda 3 detik

  // Kalibrasi Tilt
  Serial3.write(0xA5);
  Serial3.write(0x54);

  delay(1000); // Jeda sebelum kalibrasi heading

```

```
// Kalibrasi Heading
Serial3.write(0xA5);
Serial3.write(0x55);

delay(100); // Jeda sebelum konfigurasi output

// Output ASCII
Serial3.write(0xA5);
Serial3.write(0x53);

delay(100); // Jeda sebentar
}

void loop(){
}
```

2. Чтение данных

После того, как калибровка и конфигурация датчика выполнены, данные из GY25 готовы для считывания следующим образом.

```
void setup(){
  Serial.begin(115200); // Serial monitor
  Serial3.begin(115200); // Serial GY25
  delay(3000); // Jeda 3 detik

  // Kalibrasi Tilt
  Serial3.write(0xA5);
  Serial3.write(0x54);

  delay(1000); // Jeda sebelum kalibrasi heading
```

```
10
11 // Kalibrasi Heading
12 Serial3.write(0xA5);
13 Serial3.write(0x55);
14
15 delay(100); // Jeda sebelum konfigurasi output
16
17 // Output ASCII
18 Serial3.write(0xA5);
19 Serial3.write(0x53);
20
21 delay(100); // Jeda sebentar
22 }
23
24 void loop(){
25 // Baca data GY25 secara realtime
26 Serial.println(Serial3.readStringUntil('\n'));
27 }
28
```

Примечание: имейте в виду, что данные с датчика GY25 отправляются в режиме реального времени (непрерывно) в соответствии с выполненной конфигурацией. Так что показания данных также должны быть в реальном времени (нет существенной задержки). Если чтение опоздает, то данные, полученные Arduino, неточны, а если они задерживаются на долгое время, произойдет переполнение буфера.

Показания курса от датчика IMU GY25 с Arduino

Выше было рассказано о датчике GY25 и его характеристиках, а также о том, как настраивать, калибровать и считывать данные с датчика GY25

A. Курс на GY25

В этом случае я постараюсь рассказать, как получать данные о курсе (данные компаса) с датчика GY25. Получение данных о курсе с датчика GY25 имеет следующий фон

1. Датчик GY25 имеет 3 выхода одновременно, а именно Roll, Pitch и Yaw (курс), так что для использования данных датчика его нужно сначала проанализировать, то есть преобразовать строковые или двоичные данные в арифметические данные с плавающей запятой или целыми числами.
2. Данные рыскания (курса) на этом датчике имеют высокую и стабильную точность для использования в качестве эталонного компаса по сравнению с обычными датчиками компаса, такими как HMC5883L. Это связано с тем, что датчик имеет внутренний фильтр, поэтому выходные данные готовы к использованию.
3. Цена датчика GY25 очень доступна и очень надежна по сравнению с другими датчиками

B. Подготовка.

Как и прежде используем Arduino Mega, те, кто использует Arduino Uno или Nano, могут заменить его на Software Serial. Для получения данных о курсе с датчика GY25 нужно проделать следующие действия:

1. Чтобы получить высокую скорость обновления, установить скорость 115200 бод.
2. Выполнить калибровку наклона
3. Выполнить калибровку курса
4. Настроить выход в режиме реального времени ASCII.
5. Прочитать данные GY25 в режиме реального времени
6. Выполнить анализ данных, чтобы получить данные о курсе
7. Изменить данные заголовка в форме строки для плавания с помощью функции atof.

Шаг 1 - 4

Выполнение шагов 1 -4 было описанл выше. (Доступ к данным датчика GY25 с помощью Arduino).

5. Чтение данные GY25 в режиме реального времени

Поскольку данные, отправляемые с GY25, выполняются в реальном времени, чтение также должно выполняться в реальном времени, чтобы предотвратить возникновение очередей данных или даже переполнение буфера последовательных данных. Таким образом, если вы используете этот датчик в своих проектах, постарайтесь не использовать задержку delay() в вашей программе. Для задержек можно использовать функцию millis. Прежде чем читать данные датчика, нужно создать глобальную переменную для размещения данных. Ниже приведены переменные, которые мы будем использовать

```
struct gy25{
 char buffer[50];
 int counter;
 float heading;
}cmps;
```

struct используется, чтобы упростить группирование переменных одновременно, чтобы уменьшить возникновение конфликтов имен переменных. Ниже приводится объяснение каждого члена

- **buffer** : используется для отображения каждого символа, полученного от GY25
- **counter**: используется в качестве счетчика индекса символов, хранящегося в буфере
- **heading**: Используется для хранения данных о курсе в виде чисел с плавающей точкой, которые готовы к использованию для арифметических операций.

Поместите вышеуказанные переменные вне всех функций, чтобы они стали глобальными переменными, как показано ниже.

```
struct gy25{
 char buffer[50];
 int counter;
 float heading;
}cmps;
void setup(){
 Serial.begin(115200); // Serial monitor
 Serial3.begin(115200); // Serial GY25
 delay(3000); // Jeda 3 detik

 // Kalibrasi Tilt
 Serial3.write(0xA5);
 Serial3.write(0x54);

 delay(1000); // Jeda sebelum kalibrasi heading

 // Kalibrasi Heading
 Serial3.write(0xA5);
 Serial3.write(0x55);

 delay(100); // Jeda sebelum konfigurasi output

 // Output ASCII
 Serial3.write(0xA5);
 Serial3.write(0x53);

 delay(100); // Jeda sebentar
}

void loop(){
```

```
}
```

Примечание: вышеуказанная программа была добавлена с программой калибровки и конфигурации в соответствии с шагами 1 - 4

6. Анализ данных и преобразований

Перед анализом данных мы должны сначала узнать формат вывода данных с датчика GY25. Формат данных можно понять, рассмотрев следующий пример вывода данных

```
#YPR=0.00,0.01,0.02
```

- **#YPR=** : *Yaw*, *Pitch*, *Roll*
- **0.00** : data Yaw
- **0.01** : data Pitch
- **0.02** : data Roll

После пакета данных следует новая строка с символьным кодом «\ n», который является кодом ввода или новой строки. Чтобы шаги по выделению данных были следующими

1. После того, как полные данные помечены символом '\ n', то
2. Удалите первые 5 символов, а именно символы ‘#‘, ‘Y‘, ‘P‘, ‘R‘ и ‘=‘
3. Строка данных пакета - это все символы перед первым символом ‘,‘ (запятая).
4. Измените данные пакета на float

шаг 3, описанный выше, может быть выполнен с помощью функции **strtok** (String Token), которая выполняет функцию переноса строки к указанному символу. В то время как шаг 4 может быть выполнен с помощью функции **atof** (ASCII to Float).

Шаги 1 - 4 выше мы сделаем из функции updateCMPS(), чтобы ее можно было легко вызывать. Вот эта функция:

```

void updateCMPS(){
 char tmp; // Variabel temporary

 while(Serial3.available()){
  tmp = Serial3.read();
  cmps.buffer[cmps.counter++] = tmp;
  if (tmp == '\n'){ // Langkah 1
   cmps.buffer[cmps.counter] = 0; // Karakter terminator
   cmps.heading = atof(strtok(cmps.buffer+5, ",")); // Langkah 2-4
   cmps.counter = 0;
  }
 }
}

```

C. Реализация

Ниже приведен весь пример использования

```
1
2
3
4 struct gy25{
5   char buffer[50];
6   int counter;
7   float heading;
8 }cmps;
9 void setup(){
10  Serial.begin(115200); // Serial monitor
11  Serial3.begin(115200); // Serial GY25
12  delay(3000); // Jeda 3 detik
13
14  // Kalibrasi Tilt
15  Serial3.write(0xA5);
16  Serial3.write(0x54);
17
18  delay(1000); // Jeda sebelum kalibrasi heading
19
20  // Kalibrasi Heading
21  Serial3.write(0xA5);
22  Serial3.write(0x55);
23
24  delay(100); // Jeda sebelum konfigurasi output
25
26  // Output ASCII
27  Serial3.write(0xA5);
28  Serial3.write(0x53);
29
30  delay(100); // Jeda sebentar
31}
32
33void loop(){
34updateCMPS();
35Serial.println(cmps.heading); // Print heading
36}
37
38void updateCMPS(){
39char tmp; // Variabel temporary
40while(Serial3.available()){
41tmp = Serial3.read();
42cmps.buffer[cmps.counter++] = tmp;
43if (tmp == '\n'){ // Langkah 1
44cmps.buffer[cmps.counter] = 0; // Karakter terminator
45cmps.heading = atof(strtok(cmps.buffer+5, ",")); // Langkah 2-4
46cmps.counter = 0;
47}
48}
49}
50
51
52
```

Пример с использованием GY-25 и SoftwareSerial

```
//Пример чтения данных от GY-25 с использованием SoftwareSerial на ARDUINO
NANO.
//The stand-alone code for Arduino and its serial monitor (Baud=115200) is as
follows.
 ////////////////////////////////////////////////////////////////////////
 #include <SoftwareSerial.h>
 const byte rxPin = 2;        //
 const byte txPin = 3;
 /*
   The circuit:
  RX is digital pin 10 (connect to TX of other device)
  TX is digital pin 11 (connect to RX of other device)

 Note:
 Not all pins on the Mega and Mega 2560 support change interrupts,
 so only the following can be used for RX:
 10, 11, 12, 13, 50, 51, 52, 53, 62, 63, 64, 65, 66, 67, 68, 69

 Not all pins on the Leonardo and Micro support change interrupts,
 so only the following can be used for RX:
 8, 9, 10, 11, 14 (MISO), 15 (SCK), 16 (MOSI).

  */
 SoftwareSerial angleModulePort (rxPin, txPin);
 unsigned char returnBuffer[8],counter=0; //buffer where the received data
will be stored
 unsigned char fullBuffer=0; //0: buffer not full, 1: buffer is full
 int yawPitchRoll[3]; //An array of three integers. Index 0: Yaw, Index 1:
Pitch, Index 2: roll.
 unsigned char sign=0;
 unsigned char Re_buf[8];
 int YPR[3];
 // the setup function runs once when you press reset:
 void setup()
 {
 angleModulePort.begin(115200); //(115200); // initialize serial
communication between the Arduino and tilt angle module
  Serial.begin(115200); //(115200); // initialize serial communication
between the Arduino and the PC
  delay(1000);

 angleModulePort.write(0XA5); // request the data

 angleModulePort.write(0X52);
 }
 void loop()
 {
  serialEvent();
  if(sign)
 {
 sign=0;
 if(Re_buf[0]==0xAA && Re_buf[7]==0x55)
 {
 YPR[0]=(Re_buf[1]<<8|Re_buf[2])/100;
 YPR[1]=(Re_buf[3]<<8|Re_buf[4])/100;
 YPR[2]=(Re_buf[5]<<8|Re_buf[6])/100;
 Serial.println(YPR[0]);
 Serial.println(YPR[1]);
 Serial.println(YPR[2]);
  }
 }
 }
 void serialEvent()
 {
```

```
while (angleModulePort.available())
{
Re_buf[counter]=(unsigned char)angleModulePort.read();
if(counter==0&&Re_buf[0]!=0xAA) return;
counter++;
if(counter==8)
{
counter=0;
sign=1;
}
}
}
```