

Arduino и MPU6050 для определения угла наклона

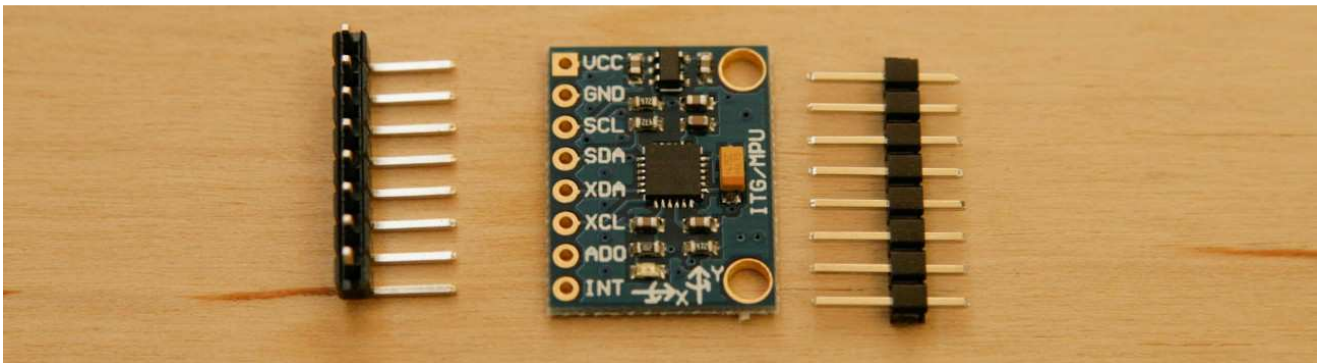
Файл заголовка (header file), приведенный ниже, будет работать с цифровыми датчиками ускорения MPU6050, которые подключены к плате Arduino через I2C протокол по адресу 0x68. Работоспособность проверена на платах Arduino Uno и Arduino Mega. Данный файл заголовка требует файл Wire.h перед добавлением "gyro_Accel.h". Кроме того, перед вызовом встроенных функций, надо инициализировать шину I2C с помощью команды Wire.begin();.

Программа для Arduino с файлом заголовка и примером расположены на https://github.com/helscream/MPU6050_Arduino

Логи версии:

Версия 0.1 beta (Дата:2014-06-22): Сам файл заголовка для калибровки и чтения данных с датчика MPU6050 через i2c протокол и пример использования заголовочного файла для расчета угла.

Версия 0.2 beta (Дата:2014-10-08): Исправлены баги в файле примера. "accel_x_scaled" и "accel_y_scaled" теперь возвращают корректные значения углов.



Глобальные переменные

Данный заголовочный файл включает в себя следующие глобальные переменные:

int accel_x_OC - Содержит измерения положения акселерометра относительно оси x при калибровке

int accel_y_OC - Содержит измерения положения акселерометра относительно оси y при калибровке

int accel_z_OC - Содержит измерения положения акселерометра относительно оси z при калибровке

int gyro_x_OC - Содержит измерения положения гироскопа относительно оси x

int gyro_y_OC - Содержит измерения положения гироскопа относительно оси y

int gyro_z_OC - Содержит измерения положения гироскопа относительно оси z

float temp_scaled - Содержит абсолютное значение температуры в градусах цельсия

float accel_x_scaled - данные оси x акселерометра минус данные калибровки

float accel_y_scaled - данные оси y акселерометра минус данные калибровки

float accel_z_scaled - данные оси z акселерометра минус данные калибровки

float gyro_x_scaled - данные гироскопа относительно оси x минус данные калибровки

float gyro_y_scaled - данные гироскопа относительно оси y минус данные калибровки

float gyro_z_scaled - данные гироскопа относительно оси z минус данные калибровки

Функции в программе Arduino для работы с mpu6050

MPU6050_ReadData()

Эта функция считывает данные с акселерометра, гироскопа и датчика температуры. После считывания данных, значения переменных (temp_scaled, accel_x_scaled, accel_y_scaled, accel_z_scaled, gyro_x_scaled, gyro_y_scaled and gyro_z_scaled) обновляются.

MPU6050_ResetWake()

Эта функция сбрасывает настройки чипа на значения по-умолчанию. Рекомендуется использовать сброс настроек перед настройкой чипа на выполнения определенной задачи.

MPU6050_SetDLPF(int BW)

Эта функция настраивает встроенный фильтр низких частот. Переменная int BW должна содержать значения (0-6). Пропускная способность фильтра будет изменяться в соответствии с представленной ниже таблицей.

int BW	Пропускная способность фильтра
0 или Any	бесконечность
1	184
2	94
3	44
4	21
5	10
6	5

Если int BW не в диапазоне 0-6, фильтр низких частот отключается, что соответствует установке – бесконечность.

MPU6050_SetGains(int gyro,int accel)

Эта функция используется для установки максимального значения шкалы измерений

int gyro	Макс. знач.[угол/с]	int accel	Макс. знач. [м/с ²]
0	250	0	2g
1	500	1	4g
2	1000	2	8g
3	2000	3	16g

MPU6050_ReadData()

Эта функция использует масштабные коэффициенты для расчета результата. Если не используются значения (0-3), MPU6050_ReadData() отобразит необработанные значения с датчика с погрешностью калибровки. Для получения обработанных значений, установите переменные для калибровки (accel_x_OC, accel_y_OC, accel_z_OC, gyro_x_OC, gyro_y_OC and gyro_z_OC) в нуль.

MPU6050_OffsetCal()

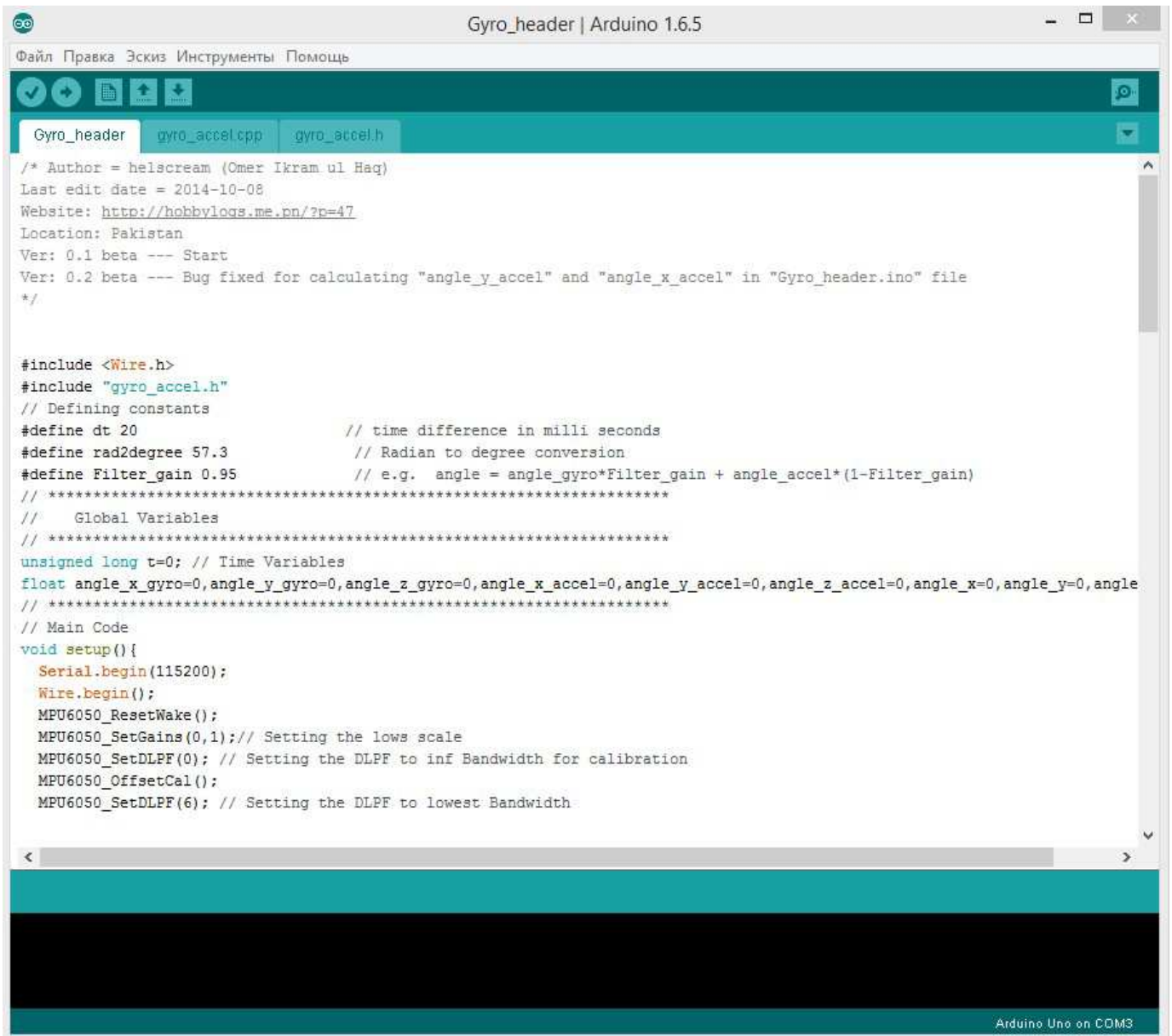
Эта функция позволяет откалибровать акселерометр и гироскоп. Рассчитанные значения записываются в переменные accel_x_OC, accel_y_OC, accel_z_OC, gyro_x_OC, gyro_y_OC и gyro_z_OC для дальнейшей коррекции. Для проведения калибровки необходимо расположить оси x и y axes платы MPU6050 в горизонтальной плоскости, а ось z – перпендикулярно к основанию. Даже незначительные перемещения платы во время калибровки понижают точность расчета базовой точки. Ось z калибруется относительно силы земного притяжения - 9.81 м/с^2 (1g), что учтено в коде.

Калибровка mpu6050

Калибровка гироскопа и акселерометра – это очень важный шаг. Приведенные значения для гироскопа имеют вид: “gyro_x_scaled = ”, так как для получения угла поворота относительно оси по данным угловой скорости, необходимо провести интегрирование. Если “gyro_x_scaled” содержит ошибку или неверно выбрана база, эта ошибка также интегрируется и превращается в значительную погрешность в результате. Так что в идеале измерения должны показывать нуль, если гироскоп не движется вокруг каких-либо осей координат. На практике добиться идеала практически невозможно, так что наша задача – минимизировать эту ошибку. Кроме того, для компенсации «дрифта», можно использовать акселерометр для расчета угла наклона, сравнения полученных данных с результатами гироскопа и последующей компенсацией данной погрешности. Расчет угла будет рассмотрен в этой статье отдельно ниже.

На рисунках далее показано использование функции *MPU6050_OffsetCal()* непосредственно в программе в Arduino IDE.

Скетч Arduino для калибровки платы акселерометра/гироскопа MPU6050:



Результат работы скетча для калибровки в серийном мониторе

```

Resetting MPU6050 and waking it up.....
The gyro scale is set to 7.63 milli Degree/s
The accel scale is set to 1.20 milli m/s^2
Calibrating gyroscope .... dont move the hardware .....
.....
gyro_x register offset = -135
gyro_y register offset = 91
gyro_z register offset = -213
Calibrating accelerometer .... dont move the hardware .....
.....
Accel_x register offset = 416
Accel_y register offset = -83
Accel_z register offset = 8538

```

Расчет угла с помощью гироскопа три6050

Данные с гироскопа имеют вид:

$$gyro_x_scaled = \frac{d}{dt} \theta_x^{gyro}$$

$$gyro_y_scaled = \frac{d}{dt} \theta_y^{gyro}$$

$$gyro_z_scaled = \frac{d}{dt} \theta_z^{gyro}$$

В дальнейшем в статье мы будем рассматривать все на примере оси x. Для расчета угла необходимо проинтегрировать переменную “gyro_x_scaled”

$$T = t_n - t_{n-1}, \text{ где } n = \{1, 2, 3, \dots\}$$

является количеством итераций

Так же стоит отметить, что на каждом временном промежутке цикла значение “gyro_x_scaled” остается одинаковым. Существует несколько подходов и методов интегрирования для компенсации и этой погрешности, но мы их детально не будем рассматривать.

Для реализации дискретного интегрирования, будем использовать метод Эйлера как один из самых популярных алгоритмов. Математически интегрирование методом Эйлера можно записать следующим образом:

$$\theta_x^{gyro}(t_n) = gyro_x_scaled * T + \theta_x^{gyro}(t_{n-1})$$

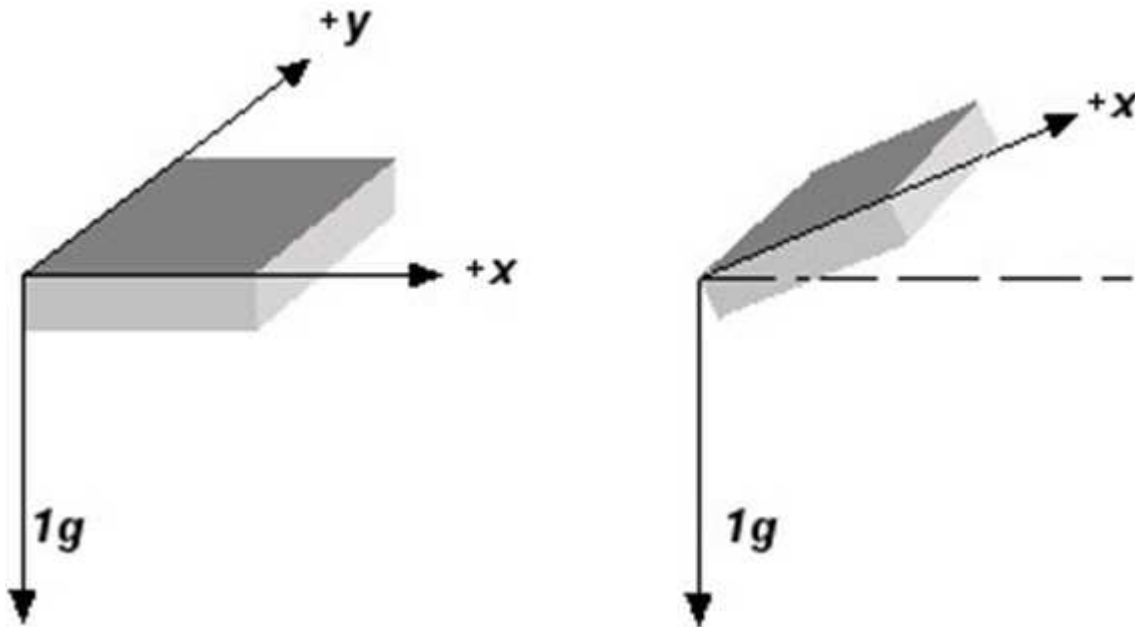
Мы предполагаем, что начальные углы относительно осей x, y, z после калибровки равны 0, 0 и 90 градусов соответственно, так что для итерации n=0:

$$\theta_x^{gyro}(t_0) = 0^\circ, \theta_y^{gyro}(t_0) = 0^\circ, \theta_z^{gyro}(t_0) = 90^\circ$$

Значение T (время каждой итерации) и динамика самого гироскопа (как быстро и насколько нелинейно изменяются углы), значительным образом влияет на точность расчетов. Чем медленнее изменяются углы и чем меньше промежуток между итерациями, тем более точным будет результат. В этом смысле жаль, что платы Arduino достаточно медленные, кристаллы у них работают с частотой 16 МГц и снятие измерений каждые 10-20 мс становится достаточно затруднительным (учитывая тот факт, что процессор занят не только расчетом угла, но и другими параллельными задачами). Мы можем использовать T в виде переменной или константы, я, лично, предпочитаю использовать константу для каждого цикла. В проекте динамические факторы не учитывались, просто использовалась частота итераций с разрывом в 20 мс (0.02 с).

Погрешность гироскопа – «дрифт» (drift)

Из-за неидеальной калибровки гироскопа, “gyro_x_scaled” никогда не равна нулю и со временем “angle_x_gyro” изменяет свои значения. Для решения данной проблемы, проводится расчет угла с помощью акселерометра и полученные значения сравниваются с углом гироскопа. Так как модуль MPU6050 располагается горизонтально, ускорение по оси z равно 1g (то есть, 9.81) как это показано на рисунке. Мы можем использовать этот вектор ускорения и его проекцию на ось y для расчета угла между осями x и y.



Угол, который рассчитывается с помощью акселерометра, рассчитывается по зависимости:

$$\theta_x^{accel} = \tan^{-1} \frac{accel_x_scaled}{\sqrt{accel_y_scaled^2 + accel_z_scaled^2}}$$

$$\theta_y^{accel} = \tan^{-1} \frac{accel_y_scaled}{\sqrt{accel_x_scaled^2 + accel_z_scaled^2}}$$

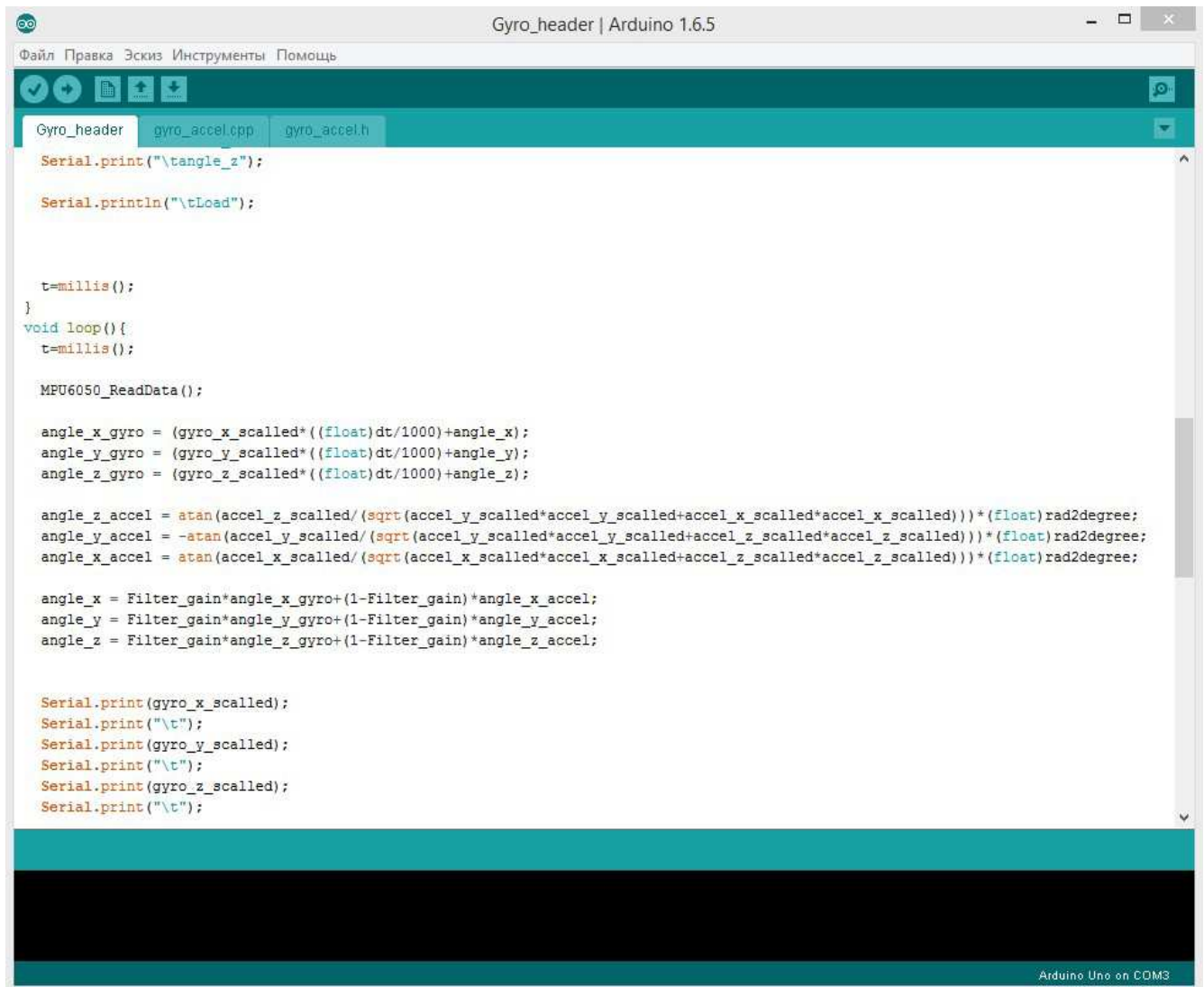
Основными проблемами при определении угла наклона с помощью акселерометра являются: сильная зашумленность сигнала и очень сильная чувствительность к вибрациям, без которых ни один механизм не работает. Более того, если при перемещении MPU6050 вдоль одной из осей координат, полученные значения будут мешать расчету угла. Так что для лучшего результата, углы с гироскопа и акселерометра объединяются с помощью фильтра:

$$\theta_x = Filter_gain * \theta_x^{gyro} + (1 - Filter_gain) * \theta_x^{accel}$$

Окончательно уравнение для определения угла наклона принимает вид:

$$\theta_x^{gyro}(t_n) = gyro_x_scaled * T + \theta_x(t_{n-1})$$

На рисунке ниже приведена имплементация полученных зависимостей в оболочке Arduino IDE



```

Gyro_header | Arduino 1.6.5
Файл Правка Эскиз Инструменты Помощь

Gyro_header gyro_accel.cpp gyro_accel.h

Serial.print("\tangle_z");

Serial.println("\tLoad");

t=millis();
}
void loop(){
t=millis();

MPU6050_ReadData();

angle_x_gyro = (gyro_x_scaled*((float)dt/1000)+angle_x);
angle_y_gyro = (gyro_y_scaled*((float)dt/1000)+angle_y);
angle_z_gyro = (gyro_z_scaled*((float)dt/1000)+angle_z);

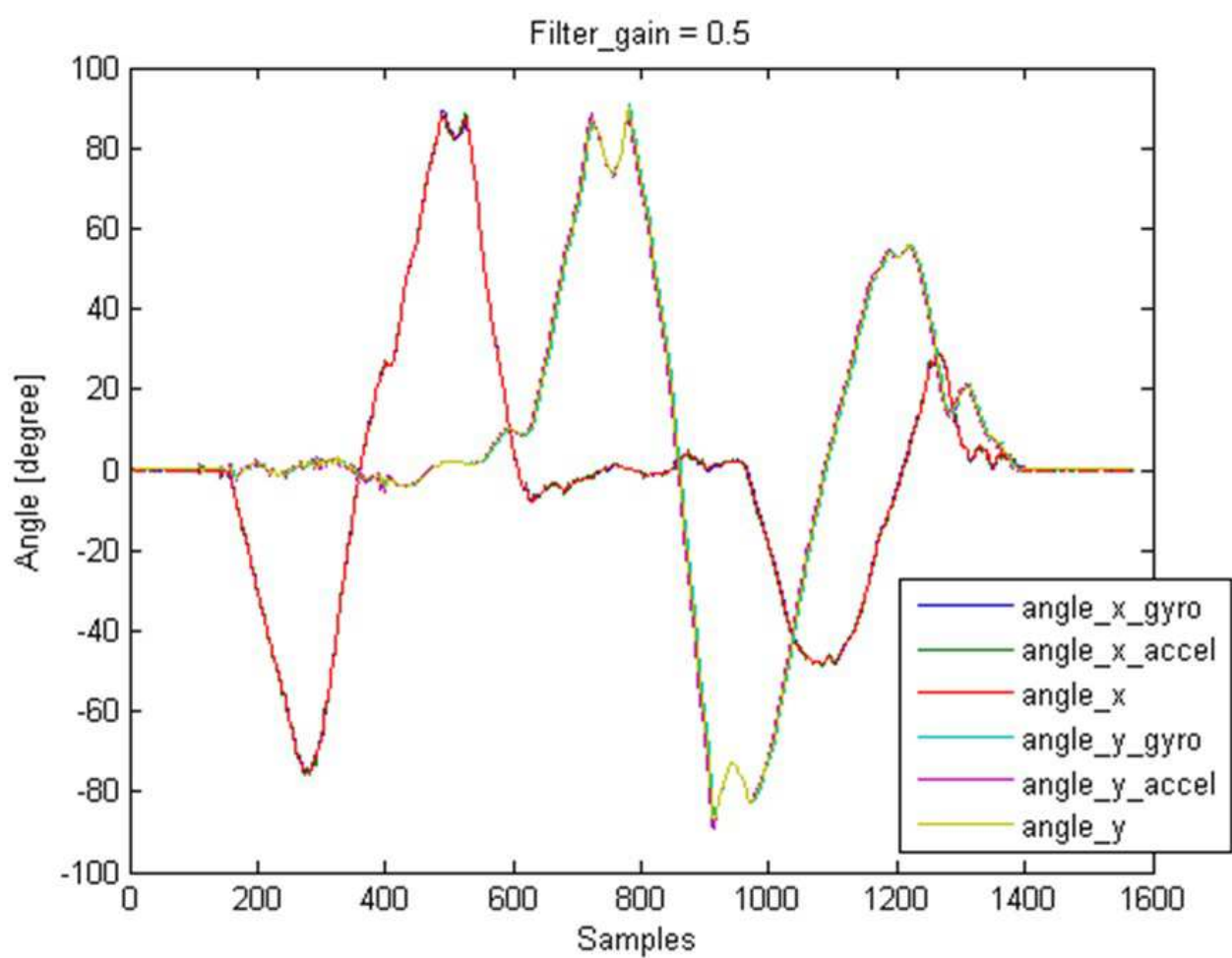
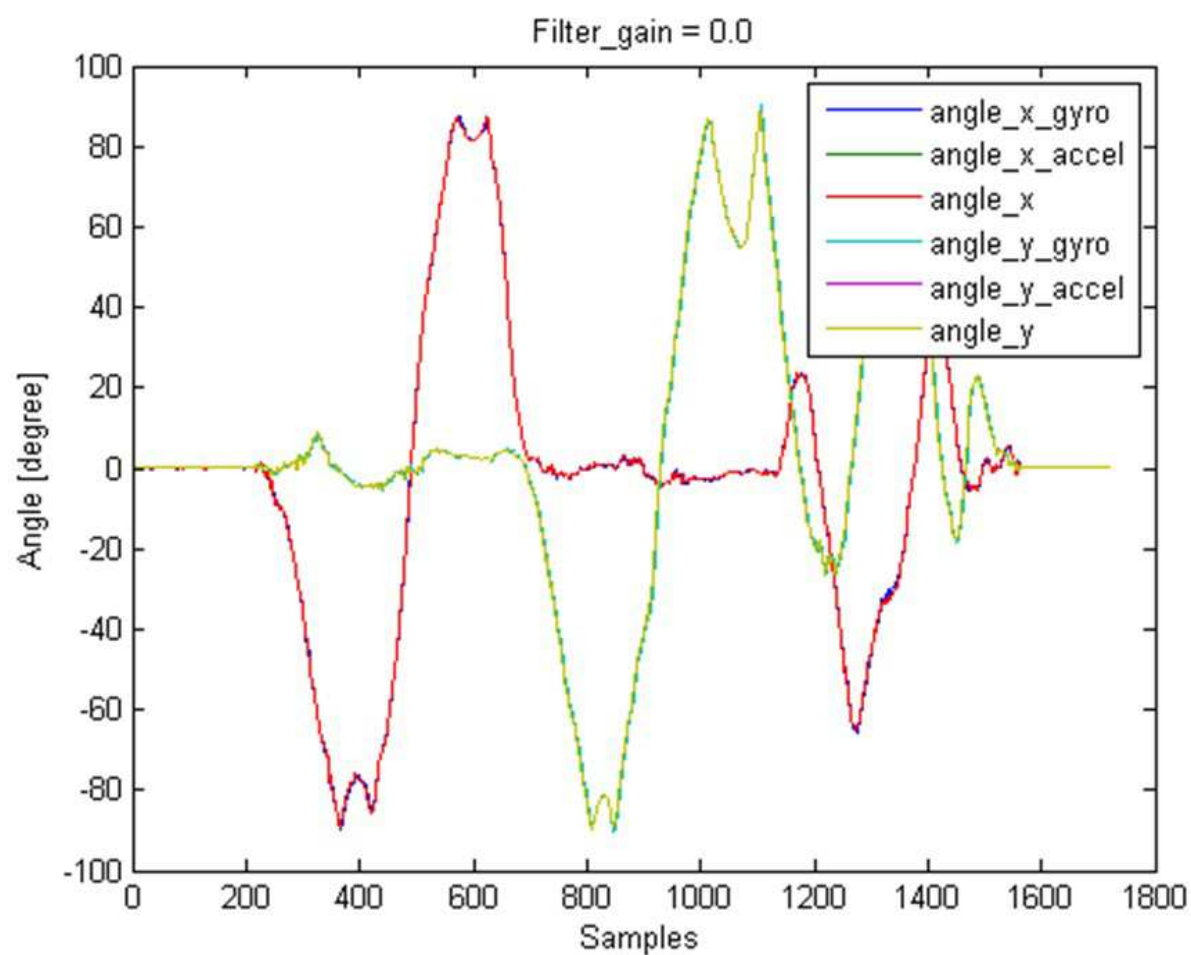
angle_z_accel = atan(accel_z_scaled/(sqrt(accel_y_scaled*accel_y_scaled+accel_x_scaled*accel_x_scaled)))*(float)rad2degree;
angle_y_accel = -atan(accel_y_scaled/(sqrt(accel_y_scaled*accel_y_scaled+accel_z_scaled*accel_z_scaled)))*(float)rad2degree;
angle_x_accel = atan(accel_x_scaled/(sqrt(accel_x_scaled*accel_x_scaled+accel_z_scaled*accel_z_scaled)))*(float)rad2degree;

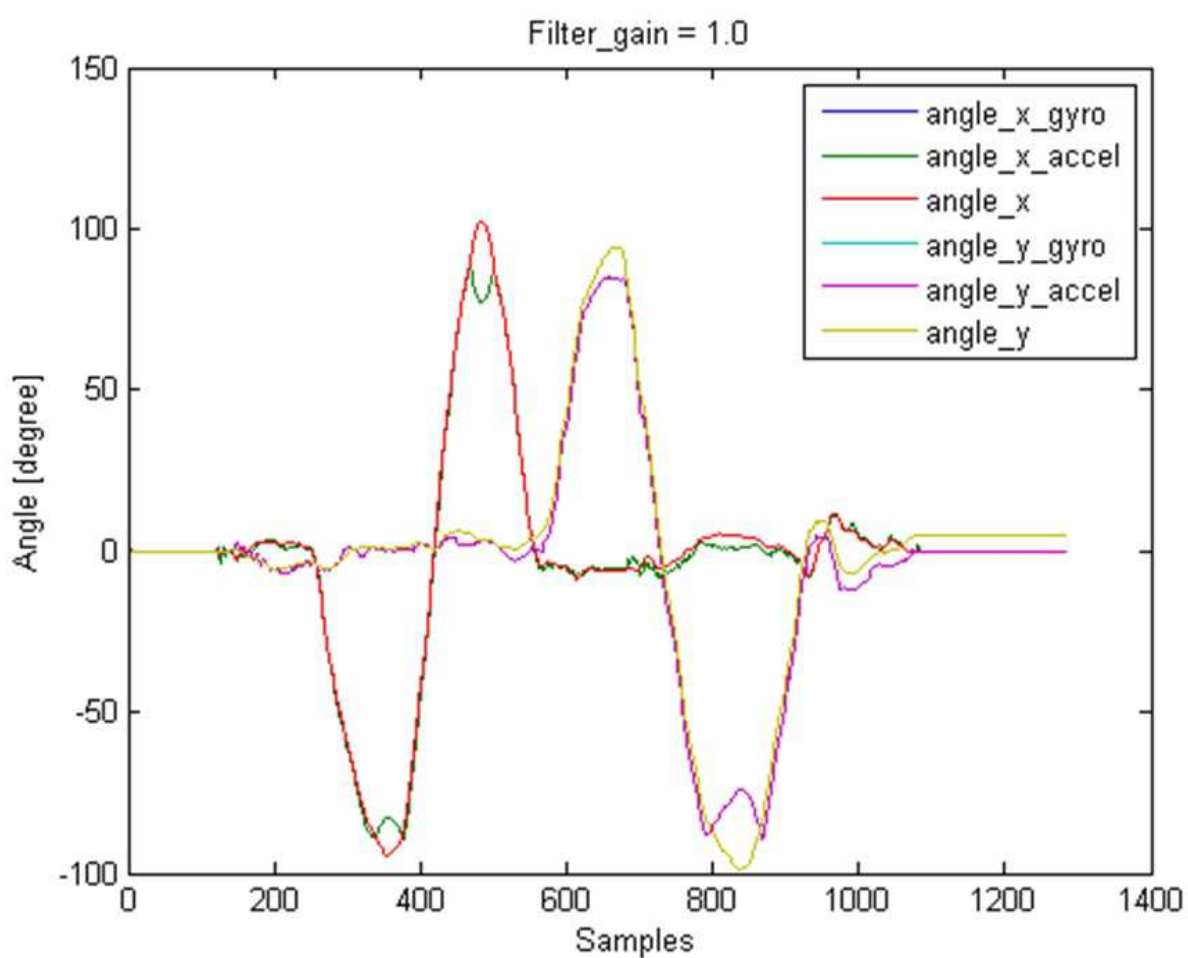
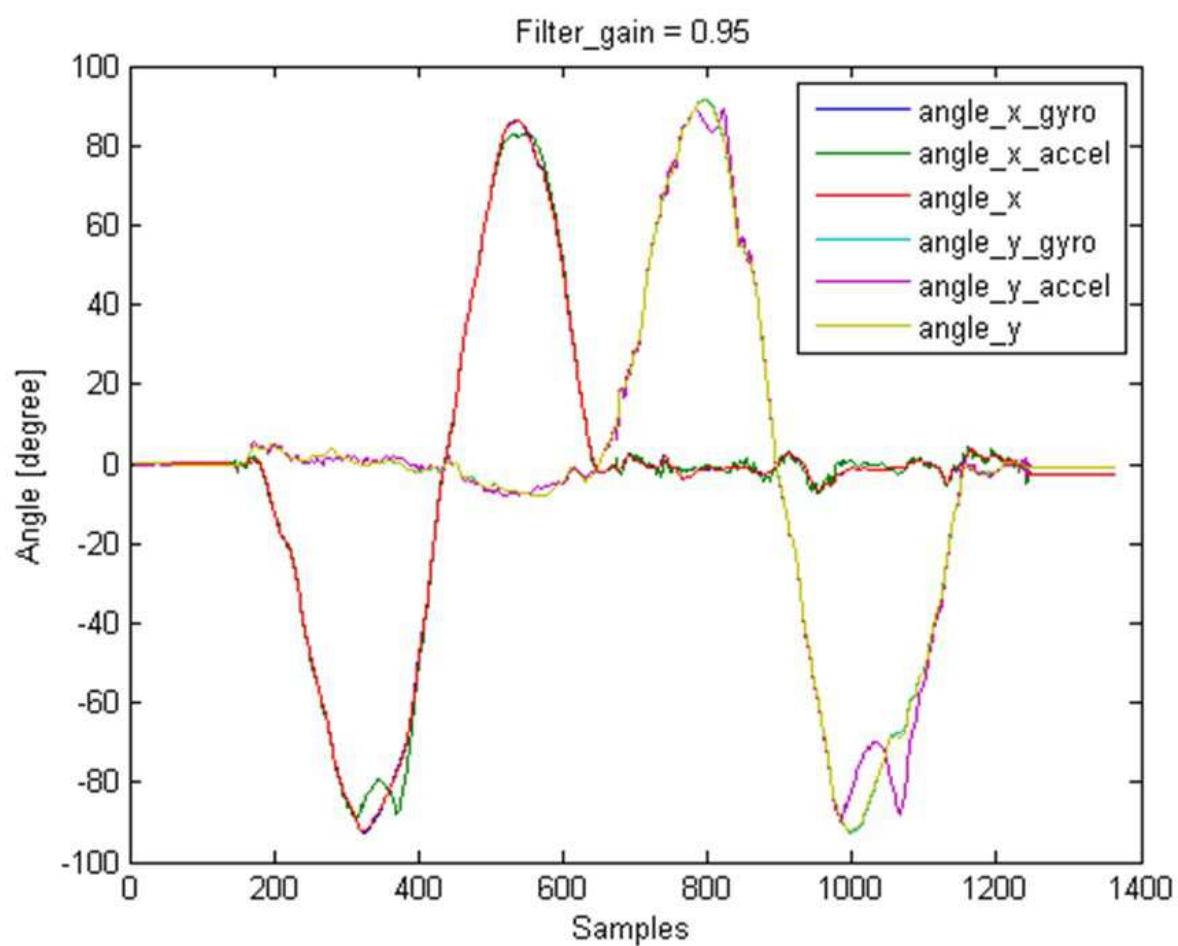
angle_x = Filter_gain*angle_x_gyro+(1-Filter_gain)*angle_x_accel;
angle_y = Filter_gain*angle_y_gyro+(1-Filter_gain)*angle_y_accel;
angle_z = Filter_gain*angle_z_gyro+(1-Filter_gain)*angle_z_accel;

Serial.print(gyro_x_scaled);
Serial.print("\t");
Serial.print(gyro_y_scaled);
Serial.print("\t");
Serial.print(gyro_z_scaled);
Serial.print("\t");
  
```

Окончательный расчет угла наклона и подбор коэффициентов усиления для фильтра

Результаты снимались для различных параметров коэффициентов усиления фильтра и приведены на рисунках по порядку. Коэффициент усиления 1 означает, что фактически идут измерения только с гироскопа. Можно заметить, что в конце angle_x и angle_y отклоняются от значений, рассчитанных с помощью значений с акселерометра.





В моем случае, для дальнейшего проекта использовался коэффициент усиления 0.95. В зависимости от динамики системы, можно его повышать, но не до 1, так как значения будут сильно отклоняться от истинных.