В конечном итоге, именно человек прописывает машине правила, как поступить в той или иной ситуации. Машине не придется выбирать, кого спасти, а кем пожертвовать. Выбирать придется человеку, который задает ей правила. Всё это похоже на классическую проблему вагонетки. Проблема вагонетки – это ситуация, когда 5 человек привязаны к рельсам и на них едет вагонетка с одним человеком внутри. Вы можете изменить направление движения вагонетки в другую сторону, на которой привязан один человек. Вам предстоит сделать выбор – убить одного человека или не делать ничего, и дать вагонетке переехать пятерых. Здесь нет правильного ответа: вы можете действовать по принципу меньшего зла или наоборот не вмешиваться.

Итак, мы поняли, что нейросети еще недостаточно развиты для обретения разума. Даже самая продвинутая нейросеть, созданная человеком, гораздо проще любого биологического мозга. Единственная реальная угроза, исходящая от искусственного интеллекта в данный момент — это массовая безработица.

Нейросеть хоть и не обладает разумом, но в решении конкретно поставленной задачи нейросеть быстро обходит человека. Так, нейросеть уже обходит человека в игре в шахматы. Роботы будут лучше человека водить машины, давать диагнозы, регулировать экономику и т.д. Неудивительно, что по проведённому исследованию BCG треть россиян бояться потерять работу из-за ИИ.

Таким образом, искусственный интеллект развит крайне слабо, чтобы говорить о его опасности для человечеств. К тому же пользы от него несоизмеримо больше даже сейчас. Боятся его определенно не стоит. Да, безработица имеет место быть, но на замену профессиям, занятых ИИ, придут новые, недоступные для интеллекта машины. Например, разработка нейросетей.

Список литературы:

1. Нейронные сети для начинающих [Электронный ресурс] / режим доступа: https://habr.com/ru/post/312450/ Дата обращения (5.04.2019)
2. Как работают нейронные сети: о сложной системе простыми словами [Электронный ресурс] / режим доступа: https://robo-hunter.com/news/kak-rabotayt-neironnie-seti-o-slojnoi-sisteme-prostimi-slovami14200 Дата обращения (6.04.2019)
3. lenta.ru [Электронный ресурс] / режим доступа: https://lenta.ru/news/2018/03/19/uberzadavil/ Дата обращения (10.04.2019)
4. Человечество в опасности: Илон Маск призвал регулировать искусственный интеллект [Электронный ресурс] / режим доступа: https://www.forbes.ru/tehnologii/347945-chelovechestvo-v-opasnosti-ilon-mask-prizval-regulirovat-iskusstvennyy-intellekt Дата обращения (15.04.2019)
5. Хокинг: искусственный интеллект - угроза человечеству [Электронный ресурс] / режим доступа:https://www.bbc.com/russian/science/2014/12/141202_hawking_ai_danger Дата обращения (10.04.2019)

ИСПОЛЬЗОВАНИЕ МИКРОКОНТРОЛЛЕРА ESP32 СОВМЕСТНО С ГИРОПРИБОРАМИ MPU-6050 И BNO055 ПРИ СОЗДАНИИ САМОБАЛАНСИРУЮЩИХ РОБОТОВ

М.В. Момот[1,a] , к.т.н., П.М. Момот[2] студенты группы 5А74
[1]Юргинский технологический институт (филиал) Национального исследовательского Томского политехнического университета,
652055, Кемеровская обл., г. Юрга, ул. Ленинградская, 26
[2]Томский политехнический университет
[a]E-mail: momotmvu@yandex.ru

Аннотация: Статья посвящена анализу возможностей электронных приборов, таких как гироскоп, акселерометр, магнитометр. Приводятся результаты тестирования и действующие алгоритмы. Анализируются возможности их применения в разных вариантах расчетов. Делаются выводы по развитию возможностей и устранению недостатков полученной комплексной системы позиционирования робота.

Abstract: Article is devoted to the analysis of opportunities of electronic devices, such as gyroscope, accelerometer, magnetometer. Results of testing and the operating algorithms are given. Possibilities of their application in different options of calculations are analyzed. Conclusions on development of opportunities and elimination of shortcomings of the received complex system of positioning of the robot are drawn.

Ключевые слова: гироскоп, акселерометр, магнитометр, BNO055, MPU5060, самобалансирующий робот.

Keyword: gyroscope, accelerometer, magnetometer, BNO055, MPU5060, the self-balancing robot.

При попытке сбалансировать в неустойчивом верхнем положении двухколесного моторного робота, которого иногда называют обратным маятником, нужно решить задачу противодействия силам, стремящимся его уронить (начать балансировать). К таким силам в первую очередь относится сила тяжести. Простейший способ балансировки заключается в анализе угла наклона робота и создании пропорционального противодействия доработкой ходовой робота.

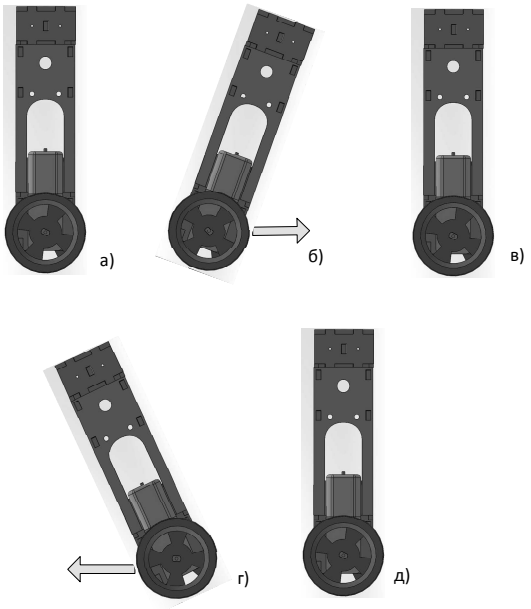

Рис. 1.Этапы балансировки обратного маятника

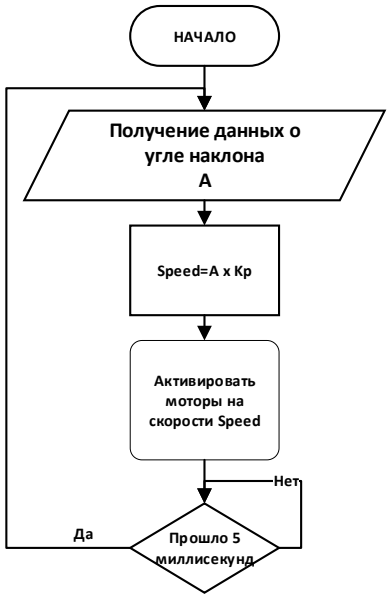

Рис. 2. Простейший алгоритм балансировки

Рассмотрим данную ситуацию на примере. На рисунке приведены варианты отклонений робота от неустойчивого равновесного состояния: Если робот из состояния а) склонился в состояние б), то следует предпринять определенное перемещение, направленное в сторону склонения, в результате чего робот, обладая инерционностью, будет возвращен в равновесное состояние. Конечно, важно не только направление противодействующего перемещения, но его скорость и продолжительность. Недостаточная скорость не позволит компенсировать скорость падения, и робот все равно упадет, тоже произойдет, если недостаточным будет длительность (об этом еще поговорим). Слишком высокая скорость и/или длительность перемещения может привести к обратному эффекту, когда робот проскочит точку равновесия и отклонится в обратную сторону (состояние г.).

Самая простая формула балансировки робота приведена ниже.

$$Speed_{motor} = A \times Kp \,, \tag{1}$$

где A – угол склонения робота (со знаком), Kp – коэффициент пропорциональности, $Speed_{motor}$ – скорость компенсирующего движения.

Для того чтобы робот не упал, следует производить коррекцию его скорости регулярно с некоторой частотой (зависит от типов двигателей и высоты центра масс робота относительно поверхности движения). В случае робота описанного ниже его корректировка производится каждые 5 миллисекунд. Алгоритм балансировки будет выглядеть как показано на рис.2. Не забываем, что угол наклона при склонении в разные стороны будет иметь разный знак, это подразумевает разное противодействующее направления вращения колес.

Последнее время, для подобных измерений используется комбинированный прибор: гироскоп, акселерометр, магнитометр BNO055, его использование значительно упрощает реализацию систем позиционирования робота, но он достаточно редок и отличается высокой стоимостью, поэтому для получения значений угла наклона робота будем рассматривать наиболее доступным на сегодняшний день комбинированный (гироскоп-акселерометр) MPU-6050.

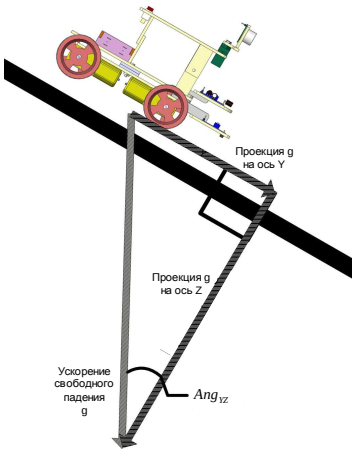

Рис. 3. Нахождение угла отклонения от вертикали (робот неподвижен)

При использовании электронного инерционного прибора MPU-6050 можно получить три вида измерений:

- угловую скорость по трем осям, которая, после математической обработки, может служить для расчета углов поворота прибора;
- проекцию вектора ускорения на три оси, которую можно применять для измерения положения прибора относительно направления вектора ускорения свободного падения;
- температуру – можно использовать для расчета корректировок показаний прибора основанных на температурных дрейфах его параметров из предыдущих пунктов.

Казалось бы, для расчета угла наклона нужно знать только показания акселерометра, т.е. берем пару проекций (зависит от того как размещен прибор на роботе) и рассчитываем угол наклона робота. Пример использования проекции ускорения свободного падения на оси Z и Y приведен на рисунке.

Для точного измерения объект должен быть неподвижен или двигаться без ускорения. В этом случае на него действует только сила притяжения, и следственно прибор измеряет только проекции ускорения свободного падения на оси акселерометра.

На рис. изображен неподвижный объект (робот) на наклонной плоскости. Координатная ось *Y* акселерометра робота направлена вперед, ось *Z* – вниз под прямым углом к наклонной плоскости, ось *X* – к наблюдателю (не используется). Искомый угол отклонения объекта от вертикали Ang_{YZ} находится по формуле:

$$Ang_{YZ} = arctg\left(\frac{a_Y}{a_Z}\right), \tag{2}$$

где a_Y – проекция ускорения свободного падения на ось Y акселерометра, a_Z – проекция на ось *Z*.

Соберем два стенда и попытаемся получить угол наклона робота.

В качестве микроконтроллера можно использовать большинство плат из семейства Arduino, мы же остановимся на наиболее симпатичной, на сегодняшний день, плате на основе контроллера ESP32. Как подключить поддержку для ESP32 в ArduinoIDE можно прочитать по ссылке http://zizibot.ru/articles/electronics/mobile-robot-on-esp32-and-shagovikakh/).

Принципиальных отличий при подключении двух различных гироприборов: MPU-6050 и BNO055 нет, оба не требовательны к питанию (хорошо себя чувствуют при напряжении 3.3в), оба подключаются к шине обмена данными I2C.

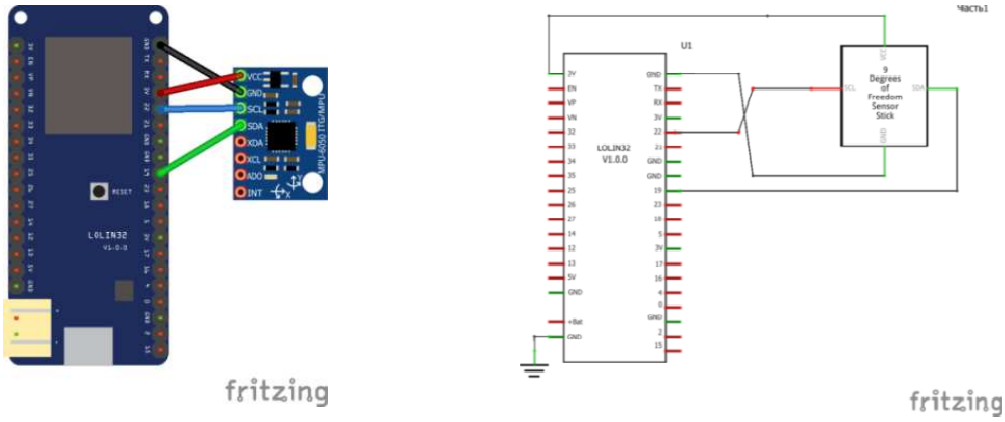

Рис. 4. Подключение MPU5060 к ESP32

Но, если MPU-6050, это только измерительный прибор, измеряющий угловую скорость и проекции вектора ускорения на оси прибора, то BNO055 содержит электронный магнитометр и встроенный микроконтроллер, который производит большинство промежуточных вычислений и может выдавать окончательные значения для использования в реальных задачах, например это могут быть углы Эйлера.

Тестирование функций BNO055 и более подробную информацию можно получить из статьи http://zizibot.ru/directory/sensor/bno055/

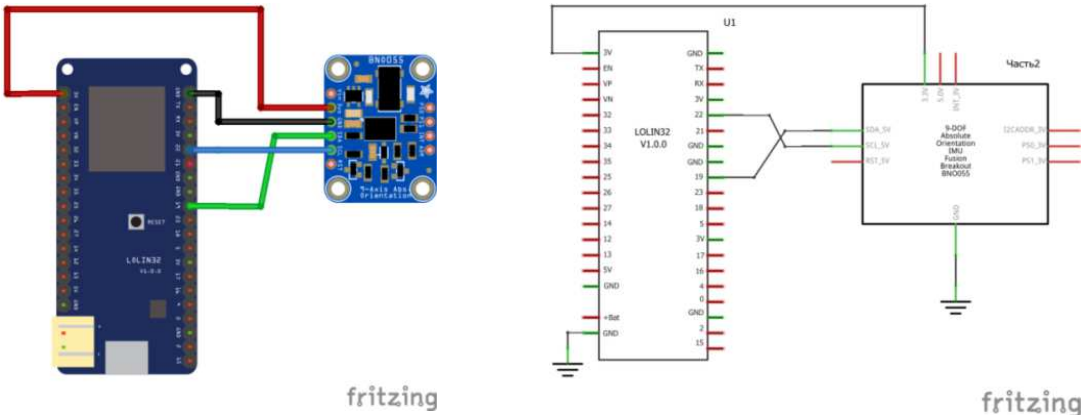

Рис. 5. Подключение BNO055 к ESP32

Так как это стенды, то отдельного питания на них подавать не нужно, оно будет поступать по usb-кабелю служащему для программирования. Кабель должен быть качественный и, по возможности новый, способный передавать токи до 500mA без нагрева, это связано с требованиями по питанию контроллера ESP32.

Для работы с MPU-6050 воспользуемся библиотекой от Korneliusz Jarzebski: https://github.com/jarzebski/Arduino-MPU6050. Я не буду устанавливать данную библиотеку в ArduinoIDE стандартными средствами, а скопирую её в каталог с тестовой программой. Потребуется заменить #include < MPU6050.h> на #include "MPU6050.h" в основной программе и файле MPU6050.cpp. Также, в связи с измененным подключением функций I2C для ESP32, нужно выполнить корректировку кода в MPU6050.cpp, как показано на рисунке.

```
#include "MPU6050.h"

bool MPU6050::begin(mpu6050_dps_t scale, mpu6050_range_t range, int mpua)
{
    // Set Address
    mpuAddress = mpua;

    Wire.begin(19,22); //Wire.begin(); //
```

Стало

Было

Рис. 6. Внесенные изменения для работы с I2C

Теперь загрузим в ESP32 следующий код, он показывает значения углов наклона, используя данные от акселерометра MPU-6050.

```ino
#include <Wire.h>
#include "MPU6050.h"
MPU6050 mpu;
void setup()
{
  Serial.begin(115200);
  Serial.println("Initialize MPU6050");
  while(!mpu.begin(MPU6050_SCALE_2000DPS, MPU6050_RANGE_2G))
  {
    Serial.println("Could not find a valid MPU6050 sensor, check wiring!");
    delay(500);
  }
}

void loop()
{
  // Read normalized values
  Vector normAccel = mpu.readNormalizeAccel();
  // Calculate Pitch & Roll
  int pitch = -(atan2(normAccel.XAxis, sqrt(normAccel.YAxis*normAccel.YAxis + normAccel.ZAxis*normAccel.ZAxis))*180.0)/M_PI;
  int roll = (atan2(normAccel.YAxis, normAccel.ZAxis)*180.0)/M_PI;
  // Output
  Serial.print(" Pitch = ");
  Serial.print(pitch);
  Serial.print(" Roll = ");
  Serial.print(roll);
  Serial.println();
  delay(50);
}
```

Рис. 7. Программа тестирования MPU-5060

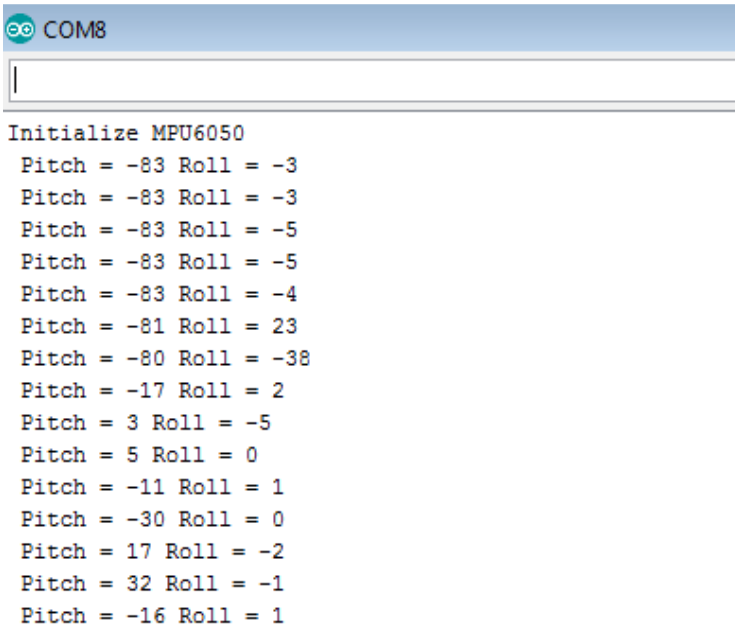

Рис. 8. Результаты работы программы

Немного изменим функцию loop, чтобы воспользоваться встроенным в ArduinoIDE плоттером последовательного соединения. Оставим только вычисления угла наклона вокруг оси Y конец строки после окончания.

```
void loop()
{
  // Read normalized values
  Vector normAccel = mpu.readNormalizeAccel();
  // Calculate Pitch & Roll
  float pitch = -(atan2(normAccel.XAxis, sqrt(normAccel.YAxis*normAccel.YAxis + normAccel.ZAxis*normAccel.ZAxis))*180.0)/M_PI;
  float roll = (atan2(normAccel.YAxis, normAccel.ZAxis)*180.0)/M_PI;
  // Output
 // Serial.print(" Pitch = ");
  Serial.println(pitch);
 // Serial.print(" Roll = ");
 // Serial.print(roll);
 // Serial.println();
  delay(5);
}
```

Рис. 9. Программа тестирования MPU-5060 адаптированная под построение графика

Анализ показывает значительные колебания полученных значений. Для нижнего рисунка я плавно поворачивал MPU-6050 вокруг оси Y, но видны сильные колебания результирующего угла, это «шум», который не позволит нам получать адекватные значения угла, используя только данные акселерометра.

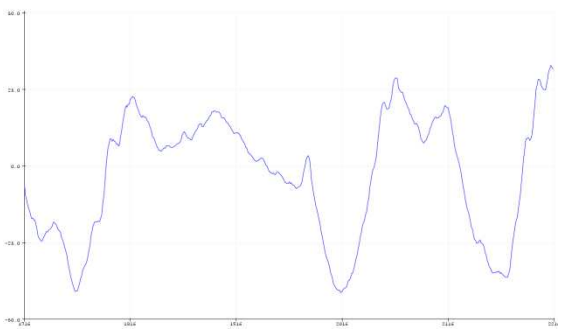

Рис. 10. Результаты работы программы в виде графика (1)

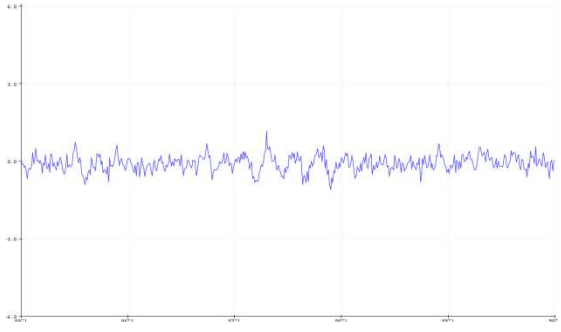

Рис. 11. Результаты работы программы в виде графика (2)

Следующий рисунок показывает, как ведет себя неподвижный акселерометр, разброс значений достигает двух градусов, что для балансирующего робота не допустимо.

```
#include <Wire.h>
#include "MPU6050.h"
MPU6050 mpu;
void setup()
{
  Serial.begin(115200);
  Serial.println("Initialize MPU6050");
  while (!mpu.begin(MPU6050_SCALE_2000DPS, MPU6050_RANGE_2G))
  {
    Serial.println("Could not find a valid MPU6050 sensor, check wiring!");
    delay(500);
  }
}
double pitch_ac = 0; double roll_ac = 0; double pitch = 0;
double roll = 0; unsigned long tx = 0; unsigned long t2 = 0; double dt;

void loop()
{
  tx = micros();
  while (t2 > tx)tx = micros();
  t2 = tx + 5000;
  dt = 0.005;
  // Read normalized values
  Vector normGyro = mpu.readNormalizeGyro();
  // Calculate Pitch & Roll
  pitch = pitch + normGyro.YAxis * dt;
  roll = roll + normGyro.XAxis * dt;
  // Output
  Serial.print(" Pitch = ");
  Serial.print(pitch);
  Serial.print(" Roll = ");
  Serial.println(roll);
}
```

Рис. 12. Программа работы с гироскопическими данными

Теперь рассмотрим, как использовать показания гироскопа. В отличие от акселерометра показания гироскопа не привязаны к определенной позиции, это только текущая угловая скорость. Для того, чтобы получить при помощи нее значение угла поворота, нужно знать предыдущее значение угла и добавить к нему изменение угла, которое есть произведение угловой скорости по выбранной оси на время в течении которого данная скорость действовала. Ниже приведен пример использования гироскопа для подсчета изменения угла.

Но, если внимательно изучить результат работы данной программы, становится очевидным, что даже при неподвижном состоянии полученные углы изменяются, этот эффект называется смещением нуля гироскопа и может быть легко устранен при помощи калибровки. В результате калибровки создаются три значения (калибровочные константы по каждой оси), которые затем отнимаются от значений, полученных от гироскопа.

За калибровку в библиотеке MPU6050.h отвечает функция calibrateGyro(). Ниже приведен пример программы с использованием калибровки. Отмечу, что после включения, стенд (MPU-6050)

в течение не менее 5 секунд должен быть неподвижен, в это время производится опрос гироскопа, и накопленная погрешность пересчитывается в качестве калибровочной. Всю программу я приводить не буду, изменения коснулись только функции setup.

```
#include <Wire.h>
#include "MPU6050.h"
MPU6050 mpu;
void setup()
{
  Serial.begin(115200);
  Serial.println("Initialize MPU6050");
  while (!mpu.begin(MPU6050_SCALE_2000DPS, MPU6050_RANGE_2G))
  {
    Serial.println("Could not find a valid MPU6050 sensor, check wiring!");
    delay(500);
  }
  mpu.calibrateGyro(1000); //
}
```

Рис. 13. Программа калибровки гироскопа

Запустите полученную программу, теперь, после калибровки, изменения угла при наклонах MPU-6050 стали много точнее, но все равно заметно накопление погрешности. К сожалению, при использовании только гироскопа, эту погрешность полностью устранить не возможно, она связана с дискретностью опроса прибора. Но, если объединить результаты работы гироскопа и акселерометра, добиться точных результатов можно.

Наиболее простым и эффективным способом объединения значений от электронного гироскопа и акселерометра является комплементарный фильтр.

$$A = 0.99(A + Asp \times dt) + 0.01 \times Aac, \quad (3)$$

где A – результирующий угол (в скобках, этот угол из предыдущей итерации расчета), Asp – текущая угловая скорость, dt – время от предыдущего опроса и расчета угла, Aac – угол, рассчитанный по показаниям акселерометра, 0.99 и 0.01 в сумме дают 1.0 и являются коэффициентами пропорциональности участия гироскопа и акселерометра в получении результата. Коэффициенты пропорциональности можно в небольшой степени изменять для достижения наилучшего результата. А ниже приведен результат работы нашей программы при плавном наклоне гироприбора – шумов нет, изменения углов происходит плавно.

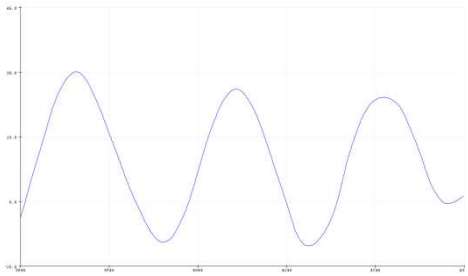

Рис. 14. Данные с гироскопа (стенд наклоняется)

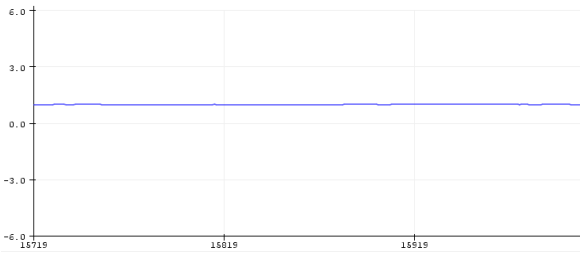

Рис. 15. Данные с гироскопа (стенд неподвижен)

А здесь гироприбор неподвижен, и наклонен на ~0.8^0. Основные измерения производятся гироскопом, а при помощи акселерометра мы устраняем дрейф нуля.

Сравнение полученного угла по данным только гироскопа (синяя линия) и при помощи комплементарного фильтра (красная линия), данных от акселерометра (зеленая линия). После нескольких секунд работы данные рассчитанные только гироскопом начинают накапливать отклонения…

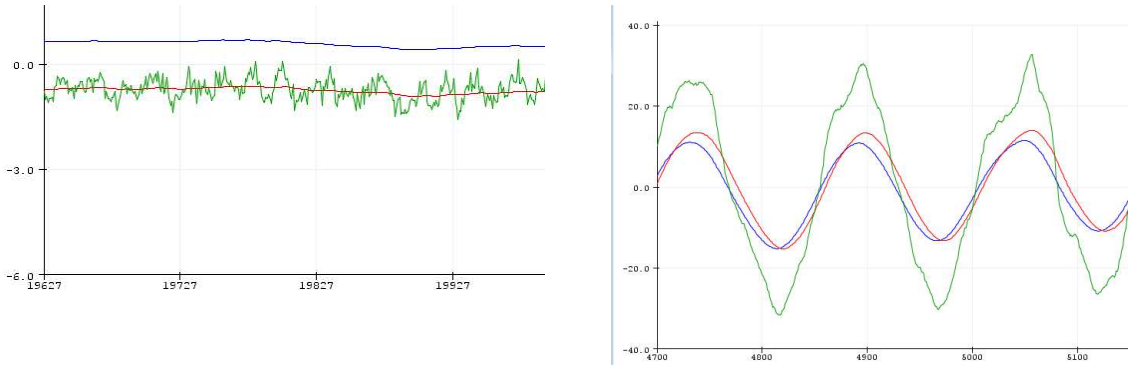

Рис. 16. Комплексные данные

Отлично!!! Мы адаптировали MPU-6050 под свои нужды, теперь рассмотрим BNO055.

Как я отметил, BNO055 содержит микроконтроллер и может сам производить расчеты, подобные тем, что мы приводили выше, когда рассматривали MPU-6050. После инициализации он начинает производить расчеты, и нам можно по необходимости, запрашивать и получать готовые результаты, например, углы наклона.

При программировании будем пользоваться библиотекой Adafruit_BNO055.h и классами, разработанными для работы с векторами см. http://zizibot.ru/directory/sensor/bno055/.

Проверим, каковы результаты запроса к BNO055 для акселерометра, если мы рассчитываем угол самостоятельно.

BNO055test1 § | Adafruit_BNO055.cpp | Adafruit_BNO055.h | imumaths.h | matrix.h | quaternion.h | vector.h

```
#include "Wire.h"
#include <Adafruit_Sensor.h>
#include "Adafruit_BNO055.h"
#include "imumaths.h"

Adafruit_BNO055 bno = Adafruit_BNO055(55,0x29);

void setup(void)
{
  Serial.begin(115200);
  Serial.println("Orientation Sensor Raw Data Test"); Serial.println("");
  if(!bno.begin())
  {
     Serial.print("Ooops, no BNO055 detected ... Check your wiring or I2C ADDR!");
    while(1);
  }
  delay(1000);
  /* Display the current temperature */
  int8_t temp = bno.getTemp();
  Serial.print("Current Temperature: ");
  Serial.print(temp);
  Serial.println(" C");
  Serial.println("");
  // bno.setExtCrystalUse(false); // Не используем внешний кварц
  bno.setExtCrystalUse(true); // используем внешний кварц
  Serial.println("Calibration status values: 0=uncalibrated, 3=fully calibrated");
}

void loop(void)
{
  // Possible vector values can be:
  // - VECTOR_ACCELEROMETER - m/s^2
  // - VECTOR_MAGNETOMETER  - uT
  // - VECTOR_GYROSCOPE     - rad/s на самом деле градусы в секунду
  // - VECTOR_EULER         - degrees
  // - VECTOR_LINEARACCEL   - m/s^2
  // - VECTOR_GRAVITY       - m/s^2
imu::Vector<3> accel = bno.getVector(Adafruit_BNO055::VECTOR_ACCELEROMETER);
  double pitch = -(atan2(accel.x(), sqrt(accel.y()*accel.y() + accel.z()*accel.z()))*180.0)/M_PI;
  double roll = (atan2(accel.y(), accel.z())*180.0)/M_PI;
Serial.println(pitch);
  delay(5);
}
```

Рис. 17. Программа работы с BNO055 (вывод данных акселерометра)

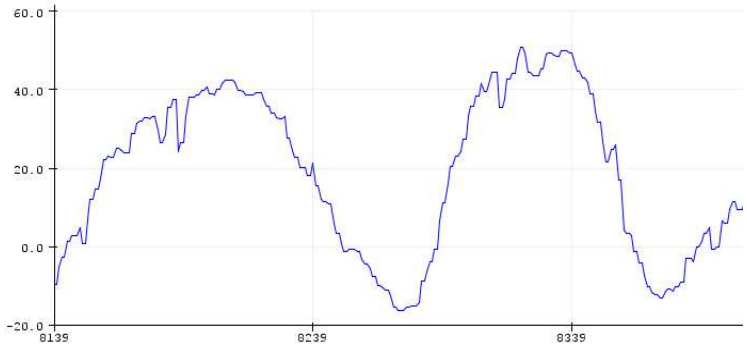

Рис. 18. Данные от акселерометра BNO055 (Визуально видно, что колебания ниже, но они есть)

Изменим программу таким образом, чтобы запрашивать не результаты работы акселерометра, а отфильтрованные по гравитации, изменив параметр вызова запроса данных от гироприбора.

```
void loop(void)
{
  // Possible vector values can be:
  // - VECTOR_ACCELEROMETER - m/s^2
  // - VECTOR_MAGNETOMETER  - uT
  // - VECTOR_GYROSCOPE     - rad/s на самом деле градусы в секунду
  // - VECTOR_EULER         - degrees
  // - VECTOR_LINEARACCEL   - m/s^2
  // - VECTOR_GRAVITY       - m/s^2
imu::Vector<3> accel = bno.getVector(Adafruit_BNO055::VECTOR_GRAVITY);
  double pitch = -(atan2(accel.x(), sqrt(accel.y()*accel.y() + accel.z()*accel.z()))*180.0)/M_PI;
  double roll = (atan2(accel.y(), accel.z())*180.0)/M_PI;
Serial.println(pitch);
  delay(5);
}
```

Рис. 19. Программа работы с BNO055 (данные фильтра BNO055)

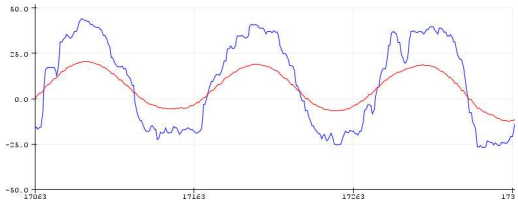

Рис. 20. Данные фильтра BNO055

Как видно из рисунка, результаты значительно лучше, но небольшие колебания присутствуют, из чего можно сделать вывод, что получение результата на основе работы комплементарного фильтра от угла акселерометра и угловой скорости гироскопа чище.

Теперь воспользуемся возможностью BNO055 по расчету углов Эйлера (крен, тангаж, рыскание). Это как раз те самые углы, расчетом которых мы занимались.

```
void loop(void)
{
  // Possible vector values can be:
  // - VECTOR_ACCELEROMETER - m/s^2
  // - VECTOR_MAGNETOMETER  - uT
  // - VECTOR_GYROSCOPE     - rad/s на самом деле градусы в секунду
  // - VECTOR_EULER         - degrees
  // - VECTOR_LINEARACCEL   - m/s^2
  // - VECTOR_GRAVITY       - m/s^2
  imu::Vector<3> accel = bno.getVector(Adafruit_BNO055::VECTOR_ACCELEROMETER);
imu::Vector<3> grav = bno.getVector(Adafruit_BNO055::VECTOR_GRAVITY);
  double pitch_a = -(atan2(accel.x(), sqrt(accel.y()*accel.y() + accel.z()*accel.z()))*180.0)/M_PI;
  //double roll_a = (atan2(accel.y(), accel.z())*180.0)/M_PI;

  double pitch_g = -(atan2(grav.x(), sqrt(grav.y()*grav.y() + grav.z()*grav.z()))*180.0)/M_PI;
  //double roll_g = (atan2(grav.y(), grav.z())*180.0)/M_PI;

imu::Vector<3> euler = bno.getVector(Adafruit_BNO055::VECTOR_EULER);

//Serial.print(-euler.y()); Serial.print(" "); Serial.println(pitch_a);
Serial.print(-euler.y()); Serial.print(" "); Serial.print(pitch_a); Serial.print(" ");  Serial.println(pitch_g);
//Serial.print(-euler.y()); Serial.print(" ");  Serial.println(pitch_g);
  delay(5);
}
```

Рис. 21. Программа (рассчитываем углы Эйлера)

Анализ значений показывает, что углы Эйлера посчитанные внутри BNO055 и углы посчитанные при помощи вектора гравитации почти совпадают. Из этого можно сделать вывод, что данные от гироскопа уже присутствуют в расчетах вектора гравитации и Эйлеровых углов, но для чистоты эксперимента составим комплементарный фильтр и на его основе рассчитаем результирующий гол наклона.

Ниже три значения: Углы Эйлера,

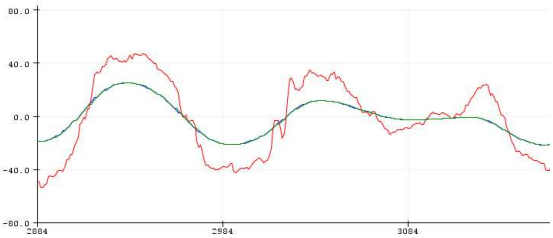

Рис. 22. Плавная линия - углы Эйлера

```
double pitch_ac = 0;
double roll_ac = 0;
double pitch = 0;
double roll = 0;
unsigned long tx = 0;
unsigned long t2 = 0;
double dt;
void loop(void)
{
  tx = micros();
  while (t2 > tx)tx = micros();
  t2 = tx + 5000;
  dt = 0.005;
  // Possible vector values can be:
  // - VECTOR_ACCELEROMETER - m/s^2
  // - VECTOR_MAGNETOMETER  - uT
  // - VECTOR_GYROSCOPE     - rad/s на самом деле градусы в секунду
  // - VECTOR_EULER         - degrees
  // - VECTOR_LINEARACCEL   - m/s^2
  // - VECTOR_GRAVITY       - m/s^2
  imu::Vector<3> accel = bno.getVector(Adafruit_BNO055::VECTOR_ACCELEROMETER);
  imu::Vector<3> gyro = bno.getVector(Adafruit_BNO055::VECTOR_GYROSCOPE); //В радианах!!!
  double pitch_ac = -(atan2(accel.x(), sqrt(accel.y() * accel.y() + accel.z() * accel.z())) * 180.0) / M_PI;
  pitch = 0.99 * (pitch + gyro.y() * dt) + 0.01 * pitch_ac;
  imu::Vector<3> euler = bno.getVector(Adafruit_BNO055::VECTOR_EULER);
  Serial.print(-euler.y()); Serial.print(" "); Serial.println(pitch);

}
```

Рис. 23. Программа (рассчитываем углы Эйлера, комплементарного фильтра)

Ниже: синяя линия расчет от BNO055 углов Эйлера, красная линия комплементарный фильтр из данных гороскопа и акселерометра.

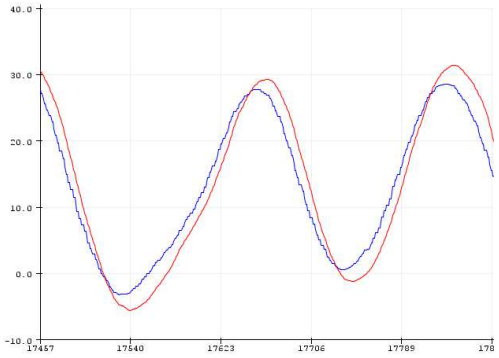

Рис. 24. Результат работы программа (углы Эйлера, комплементарного фильтра)

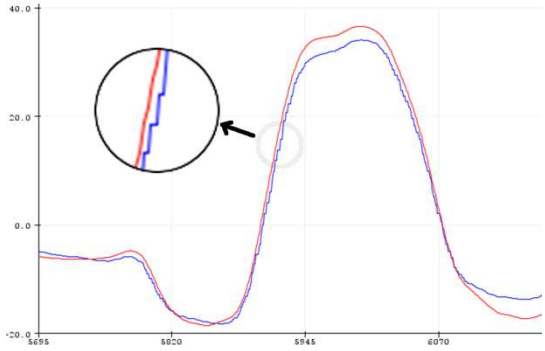

Рис. 25. Результат работы прогограмма (синяя линия – угол Эйлера рассчитанный внутри BNO055, красная линия комплементарный фильтр от данных вектора гравитации и гироскопа)

И последнее (ниже), используем вместо данных акселерометра данные вектора гравитации

```
imu::Vector<3> accel = bno.getVector(Adafruit_BNO055::VECTOR_GRAVITY);
imu::Vector<3> gyro = bno.getVector(Adafruit_BNO055::VECTOR_GYROSCOPE); //В радианах!!!
double pitch_ac = -(atan2(accel.x(), sqrt(accel.y() * accel.y() + accel.z() * accel.z())) * 180.0) / M_PI;
pitch = 0.99 * (pitch + gyro.y() * dt) + 0.01 * pitch_ac;
imu::Vector<3> euler = bno.getVector(Adafruit_BNO055::VECTOR_EULER);
Serial.print(-euler.y()); Serial.print(" "); Serial.println(pitch);
```

Рис. 26. Программа (вектор гравитации)

Синяя линия – угол Эйлера рассчитанный внутри BNO055, красная линия комплементарный фильтр от данных вектора гравитации и гироскопа. Можно видеть, что при запросе углов Эйлера встречаются ситуации, когда предыдущий и последующий (через 5-10 миллисекунд) углы совпадают, можно сделать вывод, что период в 5мсек недостаточен для гарантированного пересчета углов, и в регистрах данных BNO055 данные не успевают измениться.

Выводы. Мы изучили работу гироприборов BNO055 и MPU5060, рассмотрели особенности их использования, составили алгоритмы, которые будут использованы при построении балансирующего робота.