E4: Introduction to Programming on Arduino: I/O pins

Introduction

I/O is a short hand notation for Input/Output. As mentioned in the previous acts, the Arduino comes with various type of I/O. Namely the Analog in, and the Digital I/O. In this act, the usage of the I/O pins will be experimented using the API calls such as **digitalRead()**; **digitalWrite()**; **analogRead()**; **analogWrite()**. API stands for Application Programming Interface, a set of already defined code/function calls/methods that is usable by the programmer. http://en.wikipedia.org/wiki/Application_programming_interface
Arduino programs are written in an native language called "wiring" and it is a "C " like or "java" like code. It also uses the OOP (Object Oriented Programming) methodology. Therefore, people with prior knowledge of programming will find Arduino very easy to use. In retrospect, novice will also find it user friendly because of the simplicity of the language that focuses on design of software instead of manually write code or lost in the API jungle.
Users only need define **two functions** to make a runnable program
**setup()** – a function run once at the start of a program that can initialize settings, particularly the I/O pins settings.
**loop()** – a function called repeatedly until the board powers off.
All subsequent Arduino code written will contain these 2 functions at the minimum.

**Procedure 1**

Examine the code below from **File -> Examples -> 1.Basics -> Blink**

```
#define LED_PIN 13

void setup () {
    pinMode (LED_PIN, OUTPUT);      // enable pin 13 for digital output
}

void loop () {
    digitalWrite (LED_PIN, HIGH);   // turn on the LED
    delay (1000);                   // wait one second (1000 milliseconds)
    digitalWrite (LED_PIN, LOW);    // turn off the LED
    delay (1000);                   // wait one second
}
```

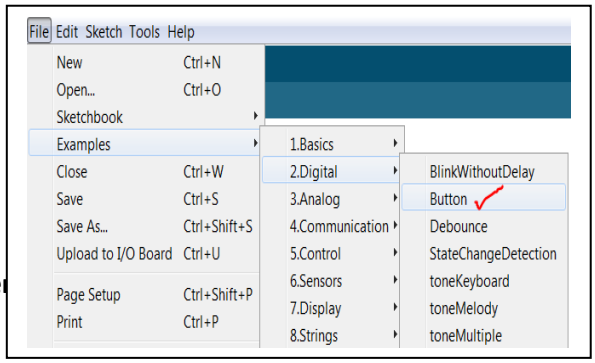
The code will first setup the parameters necessary for the Arduino project, using the setup() function. In the main body of the code, loop(), the pin is held high at +5v for 1second, and held low at 0v for 1second. Hence it is a blinking effect.

**Questions**

1. Referring to the Arduino API reference section (link from below), describe the purpose of the functions and the parameters passed to it (the items declared inside the parenthesis)
    a. pinMode(_____ , ____)
    b. delay (_____)
2. Describe the purpose of **#define LED_PIN**
3. Explain the use of "//"
4. Modify the code such that the LED will blink at different frequency. Observe the output.

**Procedure 2**

1. **Open the code from File -> Examples -> 2.Digital -> Button as per the diagram on the R.H.S.**
2. **Compile the code and download to Arduino.**
3. **Connect a LED to pin13 and ground.**
   **Connect a switch (with pull-up resistor)**
   **or a jumper wire to pin2**
4. **Assert the switch**
   **or with the other end of the wire, connect to GND and then**
   **to +5v on the Arduino. Repeat if necessary.**
5. **Observe the output at the LED.**




```
1  const int buttonPin = 2;      // the pin number of the pushbutton pin
2  const int ledPin =  13;       // the pin number of the LED pin
3  int buttonState = 0;          // variable for reading the pushbutton status
4
5  void setup() {
6    pinMode(ledPin, OUTPUT);// initialize the LED pin as an output:
7    pinMode(buttonPin, INPUT);//// initialize the pushbutton pin as an input:
8  }
9
10 void loop(){
11   buttonState = digitalRead(buttonPin);// read the state of the pushbutton value:
12   // check if the pushbutton is pressed.
13   // if it is, the buttonState is HIGH:
14   if (buttonState == HIGH) {
15     // turn LED on:
16     digitalWrite(ledPin, HIGH);
17   }
18   else {
19     // turn LED off:
20     digitalWrite(ledPin, LOW);
21   }
22 }
```

**Questions:**

1. Compare the purpose of the code at line #1 and 2 and the code from procedure1. Which line of the code describes the same purpose?
2. Referring to the Arduino API reference section (link from below), describe the purpose of the functions
   a. digitalRead(_____)

   b. if (buttonState == HIGH) {.....} else {.....}

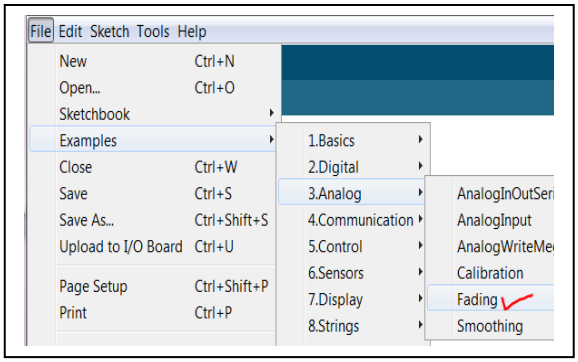
**Challenge**

1. Examine scenarios such that this piece of code and schematic could be used. List out the findings and discuss with the rest of the class.

2. Evaluate the different type of inputs that can be used with **digitalRead().**

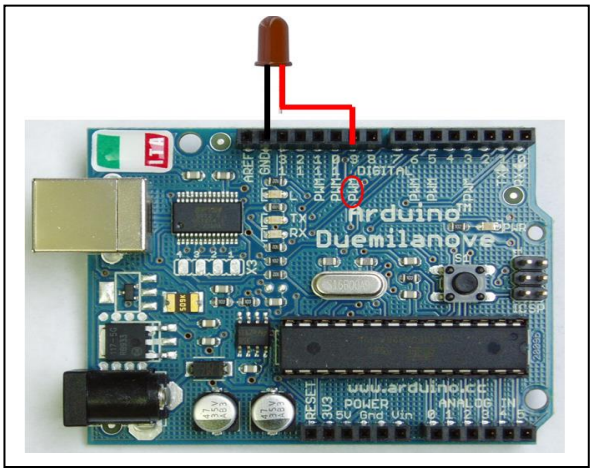
**Procedure 3**

1. **Open the code from File -> Examples -> 3.Analog -> Fading as per the diagram on the R.H.S.**
2. **Compile the code and download to Arduino.**
3. **Connect a LED to pin9 and ground. As per the diagram listed below.**
4. **Observe the output at the LED.**




```
1  int ledPin = 9;    // LED connected to digital pin 9
2  //pin3,5,6,9,10,and 11 double as PWM pin
3  void setup()  {
4    // nothing happens in setup
5  }
6  void loop()  {
7    // fade in from min to max in increments of 5 points:
8    for(int fadeValue = 0 ; fadeValue <= 255; fadeValue +=5) {
9      // sets the value (range from 0 to 255):
10     analogWrite(ledPin, fadeValue);
11     // wait for 30 milliseconds to see the dimming effect
12     delay(30);
13   }
14   // fade out from max to min in increments of 5 points:
15   for(int fadeValue = 255 ; fadeValue >= 0; fadeValue -=5) {
16     // sets the value (range from 0 to 255):
17     analogWrite(ledPin, fadeValue);
18     // wait for 30 milliseconds to see the dimming effect
19     delay(30);
20   }}
```

**Asdfasdf**




**Introduction**

analogWrite() can be used on pin3, 5, 6, 9, 10,and 11 will double as PWM pin on the Arduino. This PWM can be used to give variable output in steps of 0 to 255 (8bit MCU), which corresponds to 0v to 5v. This PWM output could be observed form the transition of the brightness of an LED and also the speed of a DC motor.

PWM (Pulse Width Modulation) is referring to a technique for producing analog outputs using digital methods (DAC). It is the inverse of the analog to digital (ADC) which will be covered in the later section. The varying pulse width generated will result in the varying analog values, when the pulses are cycle at a high frequency such as 500Hz on the Arduino.

More Info **on pg 156-157** of recommended text or at http://arduino.cc/en/Tutorial/PWM\

**Questions:**

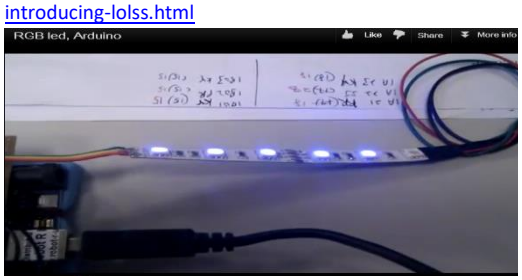
Referring to the Arduino API reference section (link from below), describe the purpose of the functions

1. analogWrite(___ , ___)
2. for (start_cond, hold_until, steps_refine) {.....}
3. Replaced the value at line #12 and #19 with 3. Justify with scientific reasoning for the value chosen in the program.
4. Evaluate the different type of outputs that can be used with analogWrite().

**Challenge** Hints: Making things talk ver2 pg 49

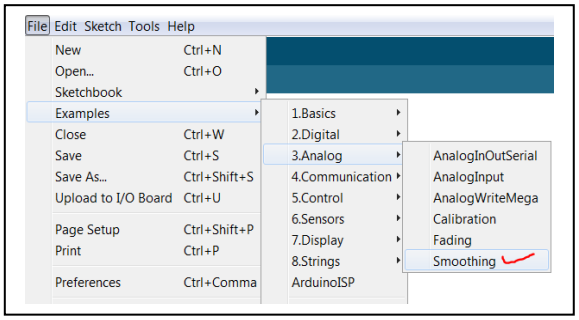
1. **Using 3 Leds' of different colour, Red, Green, Blue, and the corresponding resistors, design and build a very small project that will mix the 3 different colours. Source for a diffuser to make the light pleasant to look at.**

http://shin-ajaran.blogspot.sg/2011/11/arduino-rgb-led-introducing-lolss.html




http://shin-ajaran.blogspot.sg/2010/05/diy-arduino-and-temperature-sensor-lm35.html

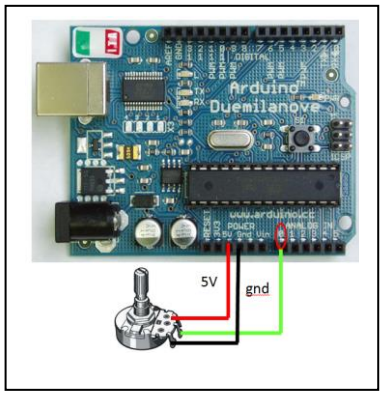
**Procedure 4**

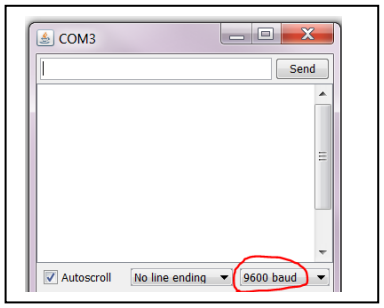



1. **Open the code from**
   **File -> Examples -> 3.Analog -> Smoothing**
   **as per the diagram on the R.H.S.**
2. **Compile the code and download to Arduino.**
3. **Connect the potentio meter as per the diagram on the lower RHS.**
4. **Click on icon for "Serial Monitor", a window will pop-up.**
5. **Observe the output as the knob is turned.**

```
const int numReadings = 10;
int readings[numReadings];      // the readings from the analog input
int index = 0;                  // the index of the current reading
int total = 0;                  // the running total
int average = 0;                // the average
int inputPin = A0;

void setup()
{
  Serial.begin(9600);// initialize for serial monitor
  // initialize all the readings in array to 0:
  for (int thisReading = 0; thisReading < numReadings; thisReading++)
    readings[thisReading] = 0;
}
void loop() {
  total= total - readings[index];// subtract the last reading
  readings[index] = analogRead(inputPin);// read from the sensor
  // add the reading to the total:
  total= total + readings[index];
  // advance to the next position in the array:
  index = index + 1;
  // if we're at the end of the array...
  if (index >= numReadings)
    index = 0;//back to the start of array
  average = total / numReadings; // calculate the average
  Serial.println(average, DEC);//send to serial monitor
}
```




**Introduction**

The code will read repeatedly from an analog input, calculating a running average and "print" the digital value that correspones to the analog signal to the serial monitor on the computer. The algorithm to achieve this is to read in the values from the sensor into an array and keep ten readings and average the readings in it, all done continuously. In other words, the MCU performs an averaging calculations to the **array** that holds several value to get a more "even out" values at the output, hence the "smoothing". What smoothing does is to eliminate the signal spike occurred in analog signals using algorithm implemented in software. This is because analog signals are very easily affected by noise, and sudden change to the analog input signal will affect the MCU to get a desired digital input signal. There is a need to implement a technique called averaging the inputs or "smoothing" in some situation.

If there is a BIG discrepancy of the value at time **t** as compared to time **t-1** and time **t+1,** value at time t is **the spike.** A spiked value that was logged will be very helpful for analysis.

Note: The higher the number of samples stored in the array for this averaging calculations, the processed input will be smoothed further, but the memory resources consumed will be even more and the output will be slower.

**Questions:**

Referring to the Arduino API reference section (link from below), describe the purpose of the functions

1. Serial.begin(___)
2. int readings[numReadings]
3. Serial.println(__)
4. Line #17 analogRead(__)
5. Purpose of line#16

**Challenge**

1. Combine the know-how from **procedure 3** with **procedure 4**. Design a schematic using simple block diagrams, modify the code and observe the output for this **very small** project. The specification is: with the turn of the knob, the colour of the LEDs' will change according to **your specifications**.
2. Explore the use of **map()** and **switch case** from the reference for "mapping" the ADC value of 0-255 to the RGB colour of Red, Green, and Blue.

Draw schematic here.

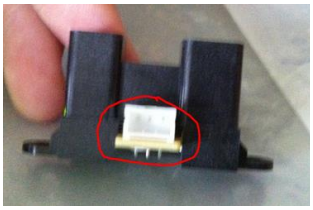
E4: Introduction to making things see on Arduino: IR sensors (Sharp GP2Y0A02)

**Introduction**

Sensors can be used to detect location or rather, sensing the physical world (analog data) and input to the MCU (digital data). The code written in the MCU could be used to make decisions and often it triggered as an output, based on the input. The sensors explored here is the Sharp Sharp GP2Y0A02, which is used to sense distance. Sensing distance is a popular application in MCU. This setup could be used to measure the distance between robots, human and robots and etc. The physical measuring parameters are different from the one used in the recommended text book Sharp GP2Y0A01. The one in the lab can detect 20cm-150cm, whereas the one in the book is only 10cm-80cm. Nonetheless, the methods to use are almost identical. An electrolyte capacitor of 10uF is usually added for this type of sensor to smoothen out the fluctuations caused by the current load. Nonetheless, it is able to function without one, as demonstrated in the exercise.

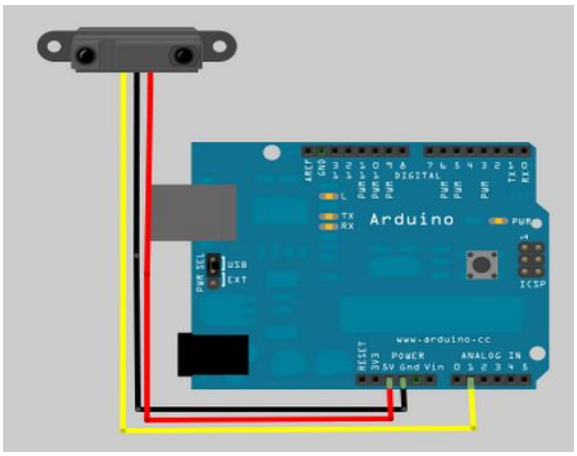
**Part List**

1x Sharp GP2Y0A02 IR sensor with jst connecors
(it could be modified from PCB headers)
1x Arduino
1x LED



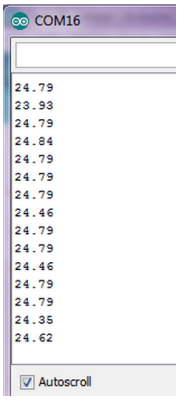
**Procedure 1**

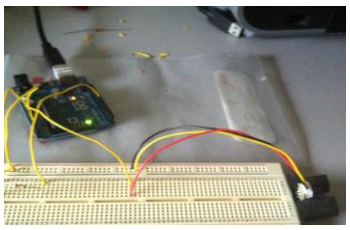
1. Refer to the GP2Y0A02 data sheet
   http://www.sharpsma.com/webfm_send/1487
   , record the electrical characteristics.
   Pay particular attention to the pin-out.
2. Connect the IR sensor accordingly, Black to gnd red to 5v and yellow to A1 on Arduino.
3. Use the code below, compile and upload to arduino.
4. Put an obstacle in front of the IR sensor and Observe and record the output from serial monitor.




Refer pg269 on making things talk v2 for connection with

```
1  int pinIR = 1; // analog pin for reading the IR sensor
2  void setup() {
3    Serial.begin(9600);// start the serial port for monitor
4  }
5
6  void loop() {
7    float volts = analogRead(pinIR)*0.0048828125;
8    // value from sensor * (5/1024), using 5v
9    float distance = 65*pow(volts, -1.10);
10   // worked out from graph 65 is
11   //theoretical distance / (1/Volts)S ,frm luckylarry.co.uk
12   Serial.println(distance);
13   // print the distance
14   delay(100);// arbitary wait time.
15
```



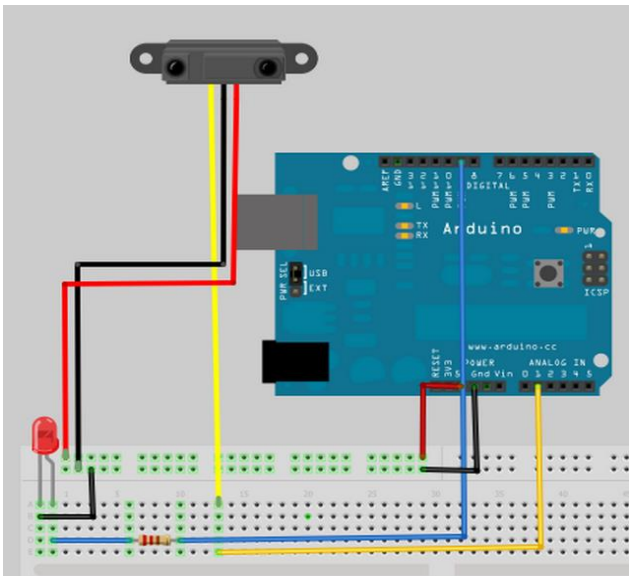



Physical connection on breadboard

Note: ADC on Arduino. The sensor is power by 5v, and the ADC is 10bits on Arduino, which yield 2^10 = 1024 divisions. In other words each of the division is 5/1024 volts could be plotted for the corresponding distance. However, the graph is not linear, from the spec sheet, the formula to determine the distance by the voltage could be determine. More info at pg 269 or here
http://luckylarry.co.uk/arduino-projects/arduino-using-a-sharp-ir-sensor-for-distance-calculation/

**Procedure 2**

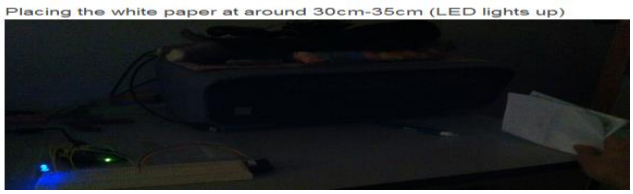
1. Repeat procedure 1 with several changes.
2. Connect the circuit as per the diagram on R.H.S
3. Connect the IR sensor accordingly with the LED and resistor. LED to pin9 and IR to pin A1.
4. Modify the code as below, compile and upload to arduino.
5. Record and observe the output.




```
int pinIR = 1;
int led = 9;


void setup() {
  pinMode(led, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  float volts = analogRead(IRpin)*0.0048828125;
  float distance = 65*pow(volts, -1.10);
  Serial.println(distance);
  delay(100);
   if (distance > 30 && distance < 40) {
    digitalWrite(led, HIGH);
  }
   else
    digitalWrite(led,LOW);
}
```

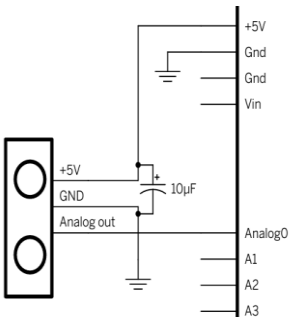




**Challenge**

Modify the code such that the intensity of the light will have a strong to dim transition with it is brightest when it is nearing the minimum distance and dimmest at the maximum distance.

**Question**

1. Study the effect of the 2 circuits with an additional 10uF capacitor. record your observation at the space given below.
2. Study the effect of using the concept of “smoothing” in this experiment. Record your observation at the space below.




pg 269 of recommended textbook

Observations

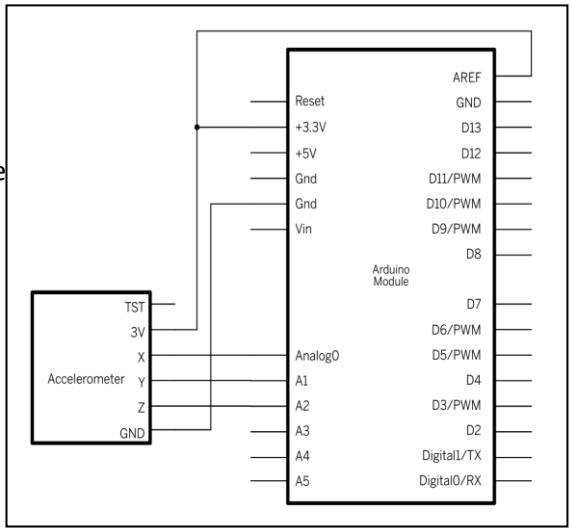
E4: Independent Research Study on the 3-axis Accelerometer (MMA7341L)

**Introduction**

The accelerometer http://en.wikipedia.org/wiki/Accelerometer was designed as part of a system for navigation and heading of an object that will have a trajectory upon propulsion. It is a very complicated piece of instruments to manufacture, prior to the manufacturing with the use of MEMS (*micro electro-mechanical systems*). This piece of instrument, implemented in MEMS was made popular in consumer electronics such as Wii and saw an alternative application besides for navigation and instrumentation, it made an in-roar development and became a de-factor feature on a number of electronic devices such as tablets, smart-phones, video-game consoles. One of the specific areas of knowledge to exploit this instrument is the gait analysis http://en.wikipedia.org/wiki/Gait_analysis.

**Deliverables**

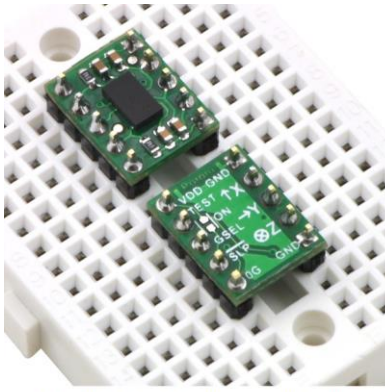
You are required to generate an **one** page report that describes the accelerometer, the usage of , how to use it and possible applications of it. For literature review, please start from pg290-293 of the recommended text book, making things talk ver2. Please note that this MMA7341L is operating on **3.3v,** and it is sensitive to **electrostatic discharge** (**ESD**). When handling the accelerometer PCB, always hold at the edge of the PCB. Do not touch the IC chip directly to prevent ESD. If unused, always keep it in and ESD protection bag (e.g the original wrapper). Refer to the Kingston installation guide on lower RHS on the preventive steps.




A sample connection can be found from the book or the diagram on the R.H.S. Also note that the pin AREF on the arduino is utilized for providing a 3.3v external reference. Hence the code must reflect this by declaring at **setup()** **analogReference(EXTERNAL);**//0-3.3v

The datasheet of this device is available at the appendix or at this URL

Your lecturer will advise on the submission date.



Two MMA7361L/MMA7341L 3-axis accelerometers plugged into a breadboard.


http://www.kingston.com/us/support/technical/installation/

**References**


1. Arduino official website, www.arduino.cc
2. http://en.wikipedia.org/wiki/Arduino
3. Arduino reference page, http://arduino.cc/en/Reference/HomePage

4. Making things talk, ver2. Tom Igoe
5. Blog, http://shin-ajaran.blogspot.sg/search/label/arduino
    a. http://shin-ajaran.blogspot.sg/2010/04/diy-how-to-start-programming-with.html
6. Element14, http://www.element14.com/community/groups/arduino
7. DFR motor shield, http://www.dfrobot.com/wiki/index.php?title=Arduino_Motor_Shield_(L298N)_(SKU:DRI0009)
8. http://www.instructables.com/id/Arduino-Motor-Shield-Tutorial/
9. List of shields, http://shieldlist.org/
10. http://arduino.cc/it/Main/ArduinoMotorShieldR3
11. MMA7341L http://www.pololu.com/file/download/MMA7341L.pdf?file_id=0J379