Digital Design W/S

Arduino 101 Bluetooth Interfacing

Tom Moxon
@PatternAgents

Intros

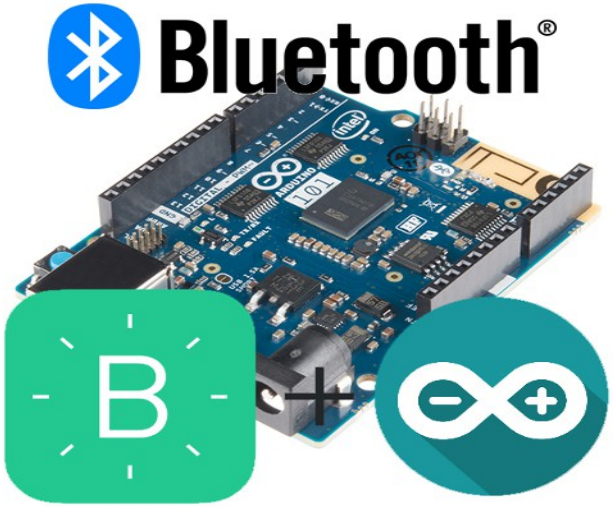
* PDX Hackerspace – Jon and Melinda
  Please donate to help support the Hackerspace,
  and ask them if you are interested in membership

* Digital Design Workshop – Tom and Stephen
  Bringing you professional grade engineering workshops
  See our calendar for upcoming workshops (Upverter)

* Hackster.io, Intel, Mikroelectronika, Newark, Element14
  Sponsored the materials and swag for the meet-up today

Instructions on Hackster.Io

* https://www.hackster.io/moxbox/arduino101-bluetooth-interfacing-3fc2bc

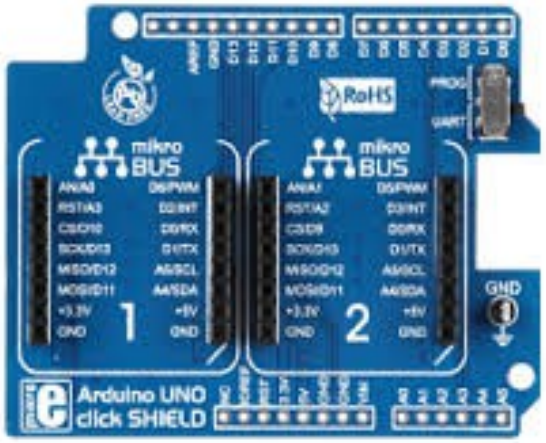


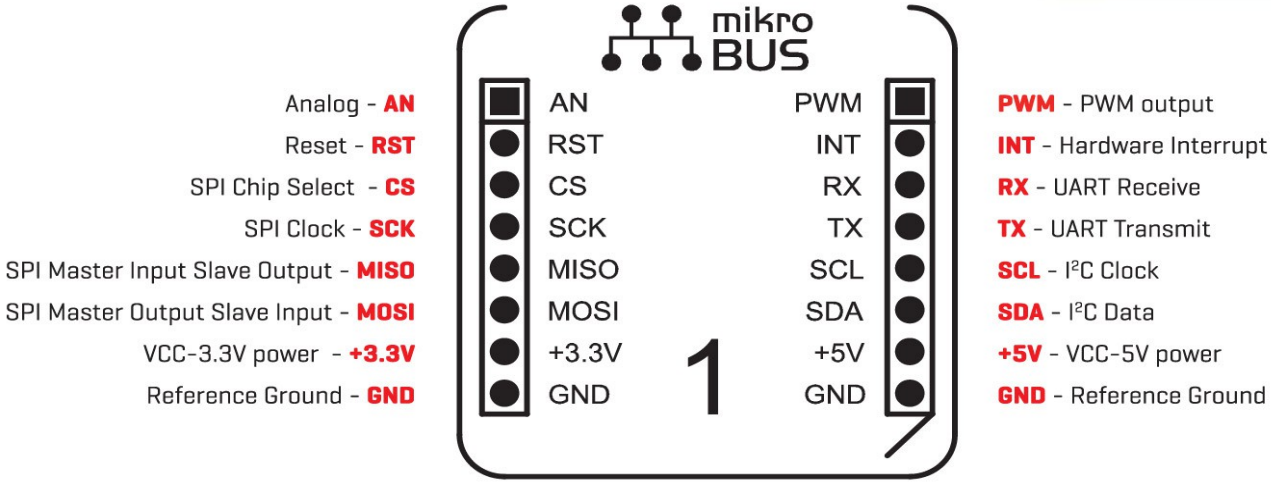
Arduino101 Platform

* Joint effort between Arduino & Intel
* The Ardunio101 has dual Intel processors X86 (Quark) and ARC core (Curie)
* Translation Libraries provide Arduino compatibility
* The Arduino101 has some built-in sensors (IMU) - 6 Axis Accelerometer & Gyroscope

Mikroelectronika

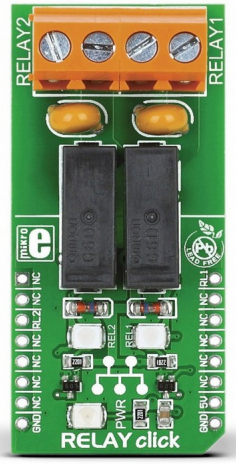
* Mikrobus Adapter
* Belgrade, Yugoslavia





Relay Click

* “Click” boards from Mikroelectronika
* All Mikrobus peripherals
* Over 250 types available (quick & easy…)
* Two (2) Mechanical relays to control higher powered lights, motors, etc.
* Be WARNED – unshielded/unprotected not recommended for 120/240 AC etc.



Intro to Arduino & Blynk

* Quick Check of experience levels here today?

* Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists and anyone interested in creating interactive objects or environments.

* Blynk is a popular mobile application for the Internet of Things. It supports a number of platforms like Arduino, ESP8266, RaspPi, and others, and runs on Android and Apple.

Arduino IDE

* What's Arduino? (Company, Board, IDE, Community?)
* Arduino abstracts many hardware details, more simple.
* Uses basic C/C++ Syntax
* Libraries extend the functionality, user contributed (big!)
* “#include wire.h” add library by including it.
  (it must be installed with Library Manager First...)

* Boards extend the supported platforms -
  make sure to use version 1.0.7 of Arduino101 boards manager
* (quick overview of the Arduinio IDE...)

Arduino Drawbacks

* No Integrated Debugger
* It's a community project, so agreement on features, etc.
* Libraries and tools are often not well tested (depends)
* Limited support of micros and platforms
* Even so – it's widely used now...

source: PatternAgents

Blynk Intro

* Started as a sucessful Kickstarter project, launched a company
* Makes creating a Graphical User Interface on smartphones as simple as “drag-and-drop” some widgets. Fast, easy.
* Bluetooth support is very new, it's a little buggy yet. (not always a Blynk problem , vendor differences, etc.)
* Wi-Fi support is pretty robust, use it for production work.
* Need to pay $$$ for extended interfaces, publishing your own applications with your own logo, etc.
* Add the concept of “virtual pins” to devices, very nice!
* Your sketch gets a “callback” from the Blynk library...

Blynk Example

* Virtual pins are named V0 thru Vxx and use a simple access method

```
BLYNK_WRITE(V1) {
  int pinData = param.asInt();
  digitalWrite(relay1_pin, pinData);
}
```

* V1 controls Relay1, V2 controls Relay2, V3 controls Relay3, V4 controls Relay4

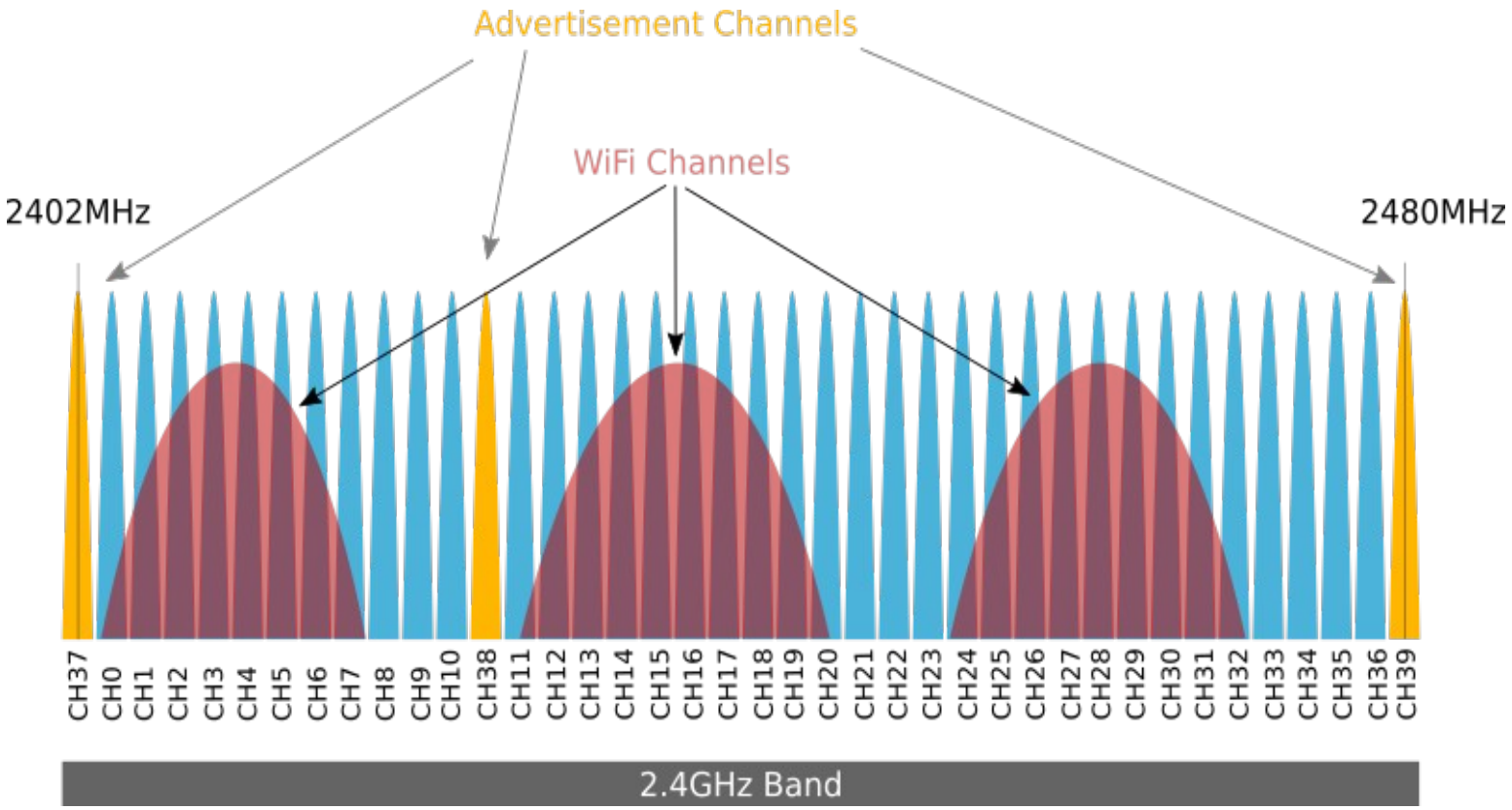
Blynk Example

Bluetooth Introduction

* A Wireless Personal Area Network
* Originally Nokia “Wibree” (2006)
* Merged into Bluetooth SIG (2010)
* Very low power transmitter
* Ubiquitous now...

Source: Wikipedia

Bluetooth vs. WiFi



Source: Bluetooth SIG

Bluetooth 4/5/LE

* Bluetooth is a rapidly expanding set of specifications
* Bluetooth Low Energy (LE) is a subset of BT 4.0 specification
* Needs newer platforms (Win8+, iOS7+, Android 4.3+, Linux 4.93+)
* BT 5.0 is adding new features like Mesh networking, Thread, etc.
* It's new, and not all devices are backward or cross compatible! (make sure you test your use cases thoroughly...)

Source: Bluetooth SIG

Bluetooth Device Roles

* GAP is an acronym for the **Generic Access Profile**, and it controls connections and advertising in Bluetooth.

* GAP defines various roles for devices, but the main two are **Central devices** and **Peripheral devices**.

* **Peripheral devices** are small, low power, resource constrained devices that can connect to a much more powerful central device. Peripheral devices are things like a heart rate monitor, a mouse, a keyboard, etc.

* **Central devices** are usually the mobile phone or tablet that you connect to with far more processing power and memory.

Source: Bluetooth SIG

Advertising and Scan Response Data

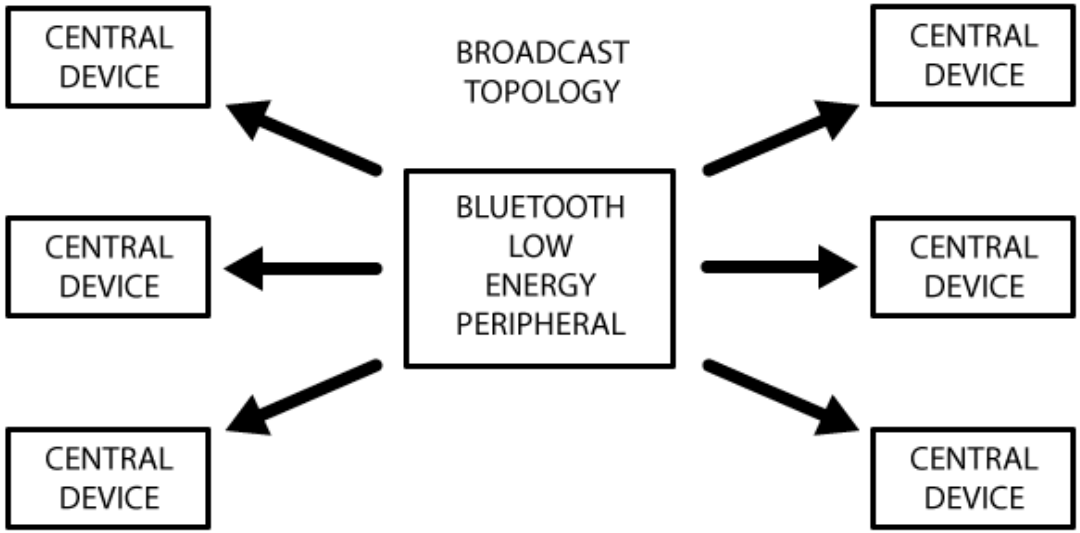
* There are two ways to send advertising data out with GAP. The Advertising Data payload and the Scan Response payload.

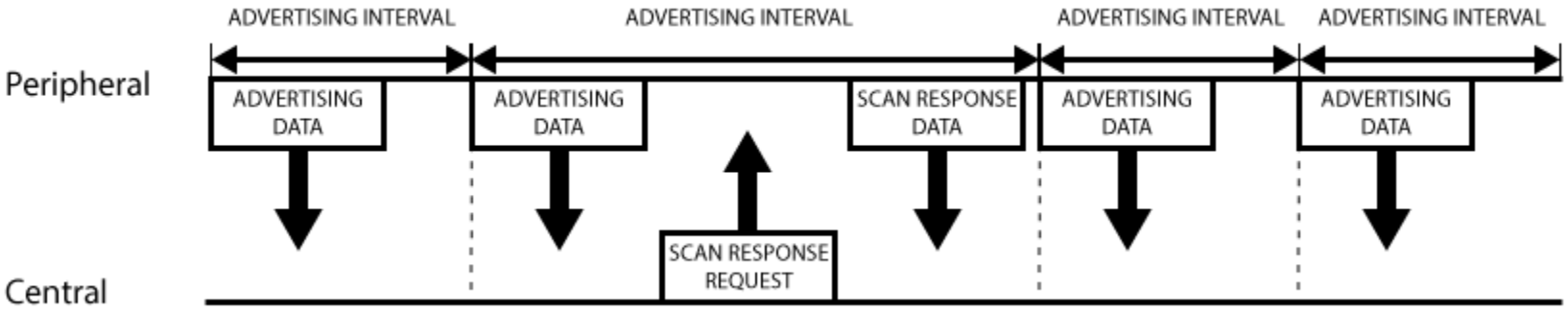
* Both payloads are identical and can contain up to 31 bytes of data, but only the advertising data payload is mandatory, since this is the payload that will be constantly transmitted out from the device to let central devices in range know that it exists.

* The scan response payload is an optional secondary payload that central devices can request, and allows device designers to fit a bit more information in the advertising payload such a strings for a device name, etc.

Source: Bluetooth SIG

Bluetooth Network Topology





Source: Bluetooth SIG

GATT and Profiles

* **GATT** is an acronym for the Generic Attribute Profile, and it defines the way that two Bluetooth Low Energy devices transfer data back and forth using concepts called Services and Characteristics.
* It makes use of a generic data protocol called the Attribute Protocol (ATT), which is used to store Services, Characteristics and related data in a simple lookup table using 16-bit IDs for each entry in the table.
* **GATT** comes into play once a dedicated connection is established between two devices, meaning that you have already gone through the advertising process governed by **GAP**.

Source: Bluetooth SIG

GATT Transactions

* The peripheral is known as the GATT Server, which holds the ATT lookup data and service and characteristic definitions, and the GATT Client (the phone/tablet), which sends requests to this server.

* All transactions are started by the master device, the GATT Client, which receives response from the slave device, the GATT Server.

* When establishing a connection, the peripheral will suggest a 'Connection Interval' to the central device, and the central device will try to reconnect every connection interval to see if any new data is available, etc. It's important to keep in mind that this connection interval is really just a suggestion, though! Your central device may not be able to honour the request because it's busy talking to another peripheral or the required system resources just aren't available.

Source: Bluetooth SIG

Profiles and Services

* **Profile** - a pre-defined collection of Services that has been compiled by either the Bluetooth SIG or by the peripheral designers. The Heart Rate Profile, for example, combines the Heart Rate Service and the Device Information Service.

* **Services** are used to break data up into logical entities, and contain specific chunks of data called characteristics. A service can have one or more characteristics, and each service distinguishes itself from other services by means of a unique numeric ID called a UUID, which can be either 16-bit (for officially adopted BLE Services) or 128-bit (for custom services).

Source: Bluetooth SIG

Characteristics

* The lowest level concept in GATT transactions is the **Characteristic**, which encapsulates a single data point (though it may contain an array of related data, such as X/Y/Z values from a 3-axis accelerometer, etc.).

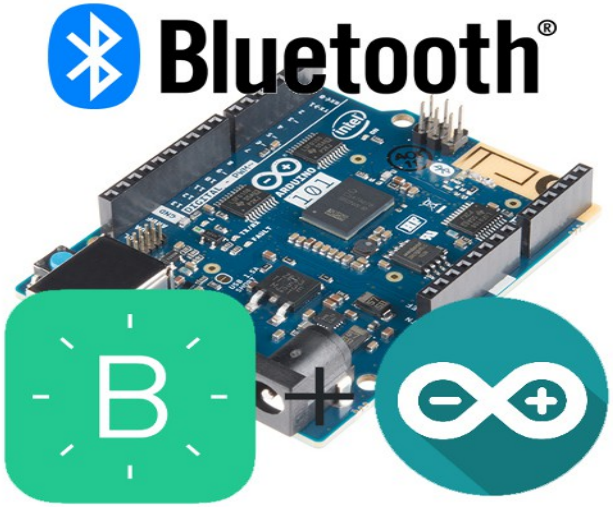
* Similarly to Services, each Characteristic distinguishes itself via a pre-defined 16-bit or 128-bit UUID, and you're free to use the standard characteristics defined by the Bluetooth SIG or to define your own custom characteristics which only your peripheral and SW understands.

* As an example, the Heart Rate Measurement characteristic is mandatory for the Heart Rate Service, and uses a UUID of 0x2A37. It starts with a single 8-bit value describing the HRM data format (whether the data is UINT8 or UINT16, etc.), and the goes on to include the heart rate measurement data that matches this config byte.

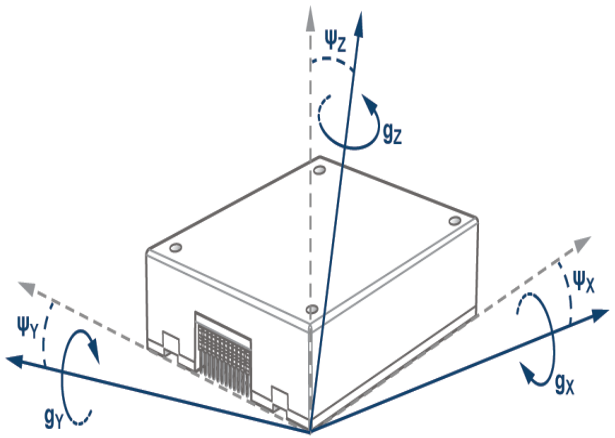
Source: Bluetooth SIG

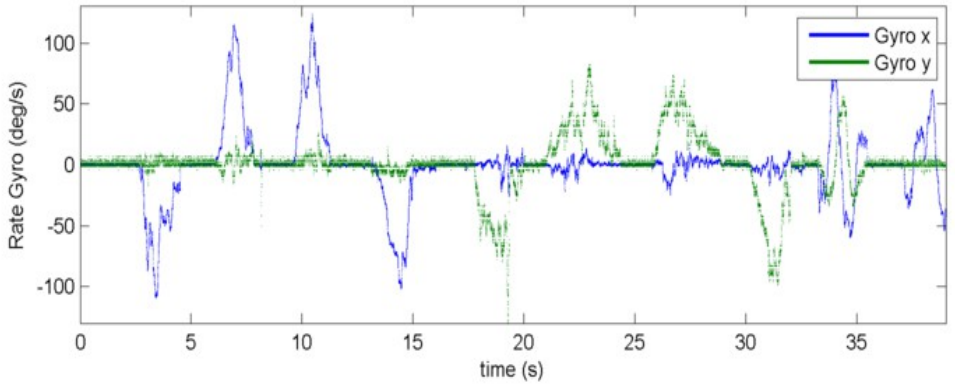
Instructions on Hackster.Io

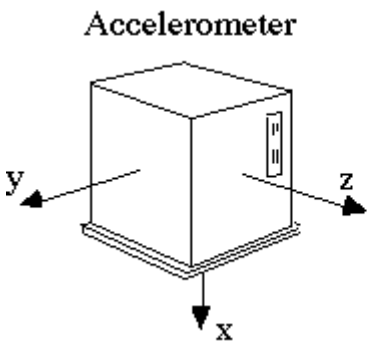
* https://www.hackster.io/moxbox/arduino101-bluetooth-interfacing-3fc2bc

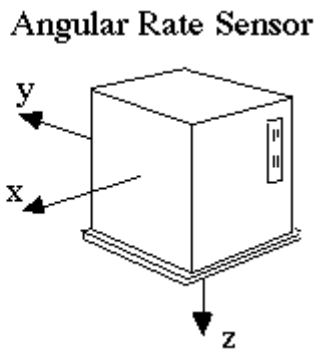


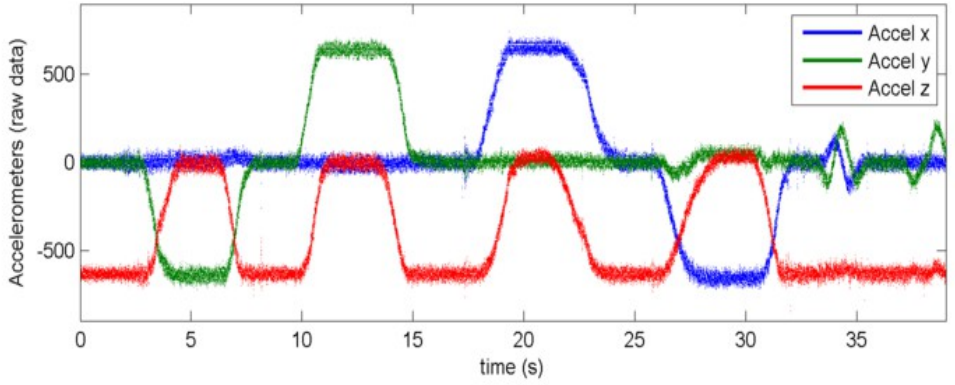
Gyro and Accelerometer











CurrieIMU Library

* begin()
* GetGyroRate()
* GetAccelerometerRate()
* getAccelerometerRate()
* setAccelerometerRate()
* getGyroRange()
* setGyroRange()
* getAccelerometerRange()
* setAccelerometerRange()
* autoCalibrateGyroOffset()
* autoCalibrateAccelerometerOffset()
* noGyroOffset()
* noAccelerometerOffset()
* gyroOffsetEnabled()
* (much more…)

CurrieIMU Setup

```
#include "CurieIMU.h"

void setup() {
  Serial.begin(115200); // initialize Serial communication
  while (!Serial);    // wait for the serial port to open

  // initialize device
  Serial.println("Initializing IMU device...");
  CurieIMU.begin();

  // Set the accelerometer range to 250 degrees/second
  CurieIMU.setGyroRange(250);
 // Set the accelerometer range to 2G
  CurieIMU.setAccelerometerRange(2);
}
```

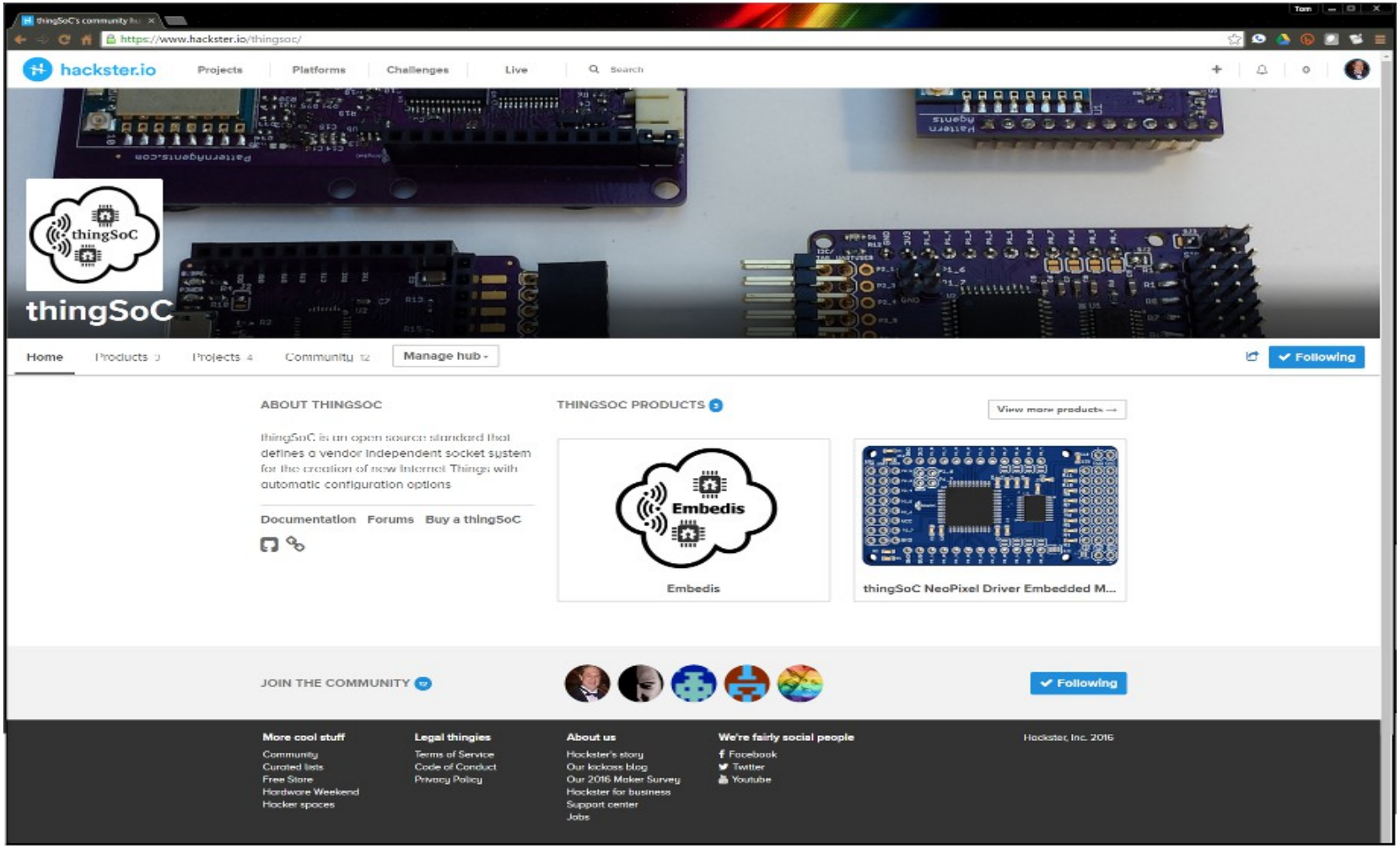
CurrieIMU Gyro

```
void loop() {
  float gx, gy, gz; //scaled Gyro values

  // read gyro measurements from device, scaled to the configured range
  CurieIMU.readGyroScaled(gx, gy, gz);

  // display tab-separated gyro x/y/z values
  Serial.print("g:\t");
  Serial.print(gx);
  Serial.print("\t");
  Serial.print(gy);
  Serial.print("\t");
  Serial.print(gz);
  Serial.println();
}
```

CurrieIMU Accelerometer

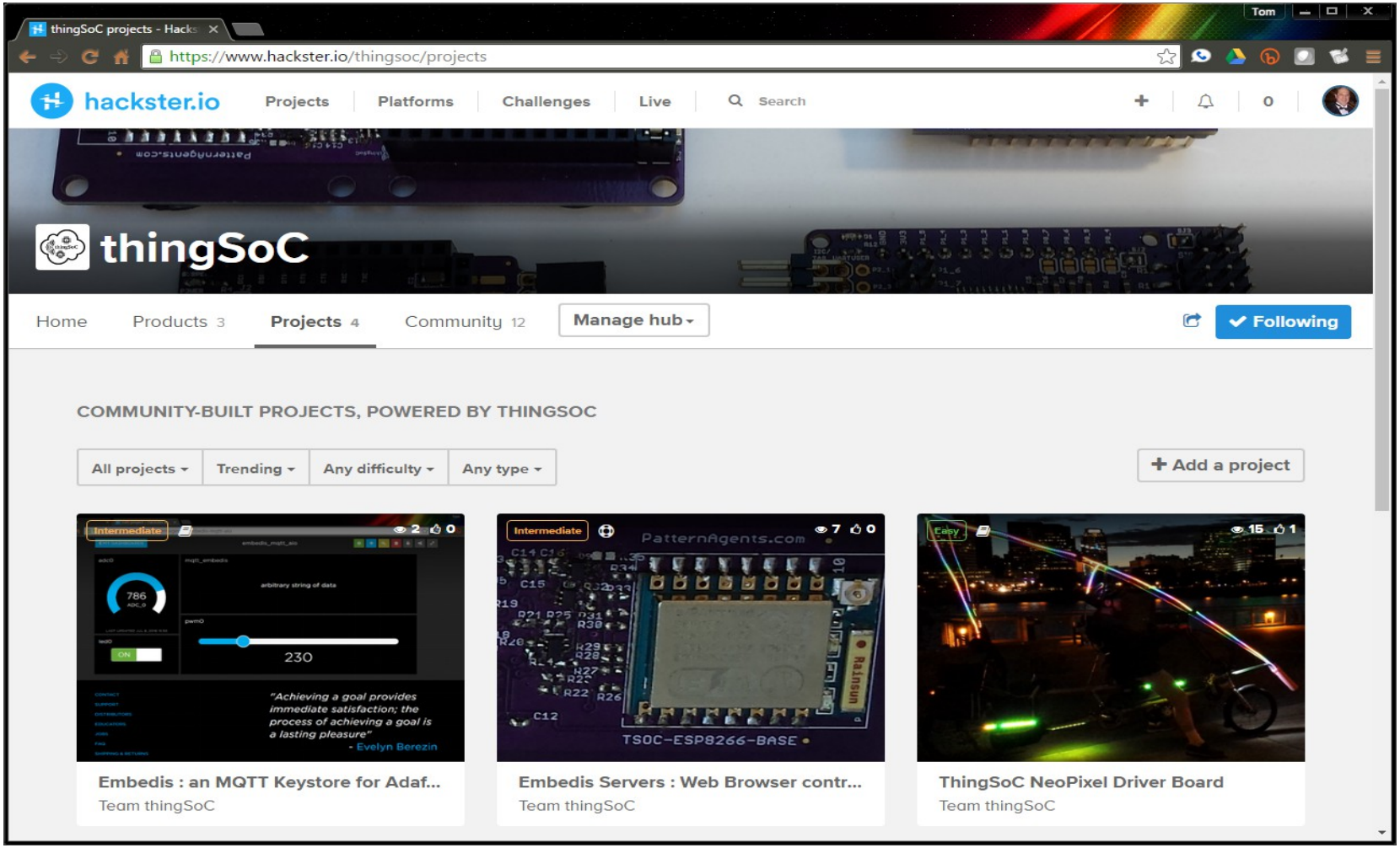
```
void loop() {
  float ax, ay, az;   //scaled accelerometer values

  // read accelerometer measurements from device, scaled to the configured range
  CurieIMU.readAccelerometerScaled(ax, ay, az);

  // display tab-separated accelerometer x/y/z values
  Serial.print("a:\t");
  Serial.print(ax);
  Serial.print("\t");
  Serial.print(ay);
  Serial.print("\t");
  Serial.print(az);
  Serial.println();
}
```

Hackster.io Platform



source: PatternAgents

Hackster.io Projects



source: PatternAgents

If you would like to help...

* Sign Up for Hackster.IO
* Please go to :
  https://www.hackster.io/thingsoc
  and “Click” on the blue “Following” button
* We'll be having some thingSoC workshops in June!

Thank You !

Thank You!

Source : @PatternAgents