Raspberry Pi HW/SW Application Development

Make your own shield and control peripheral using RPi

Walter Dal Mut & Francesco Ficili

Intro to RPi

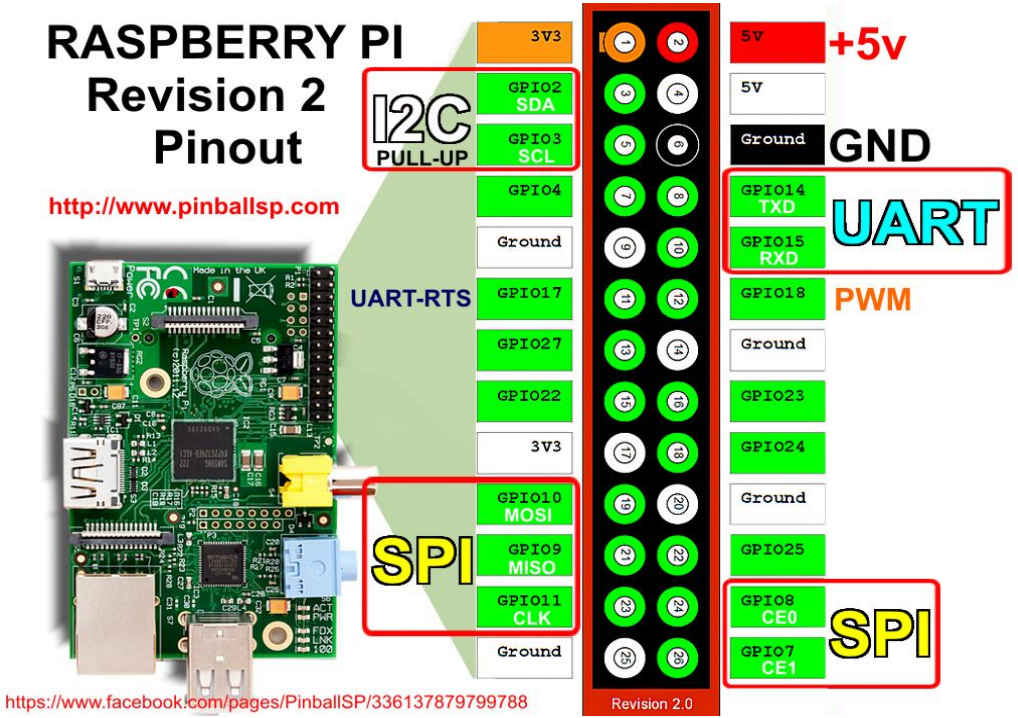
- Basically a Microcomputer on a ‘Credit Card sized’ board
- Based on a Broadcom SoC
- Many different interfaces: USB, Ethernet, HDMI, SD, TFT Display...

Model B

Model B+

...And a Mysterious Thing

- The RPi GPIO (General Purpose IO) Connector contains all low-level interface of the board.
- Usually not known by high-level software programmers
- Well known by embedded software developers.
- It’s the best choice to make board functionalities expansions.

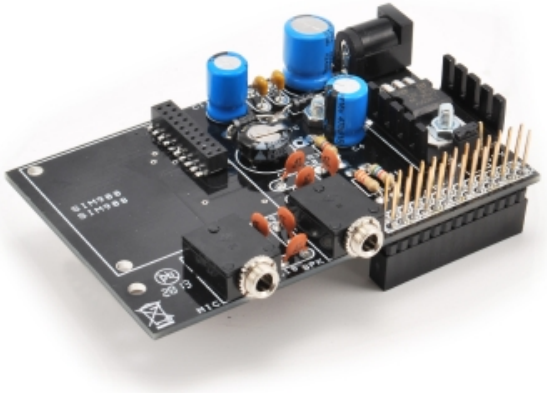


Interface Options

- GPIO: the General Purpose Input Output is the most simple interface of the board. It can writes or reads the status of a single pin.
- UART: the Universal Asyncronous Receiver Transmitter is one of the most used serial standard in data communication. It's the TTL level version of the old PC RS232 interface (tty under Linux). It is a full-duplex point-to-point standard.
- I2C: the Inter Integrated Circuit (aka I square C) is a communication standard developed by Philips during '80. It's a address oriented, half-duplex, master-slave multidrop standard. It also support a multi-master variant.
- SPI: the Serial Peripheral Interface is a communication standard developed by Motorola. It is a master-slave multidrop full-duplex standard.

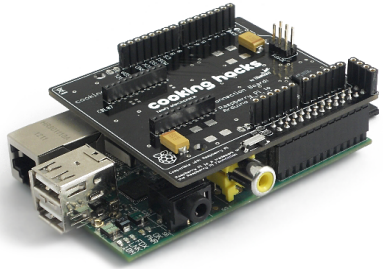
The ‘Shield’ Concept

- Shield Definition (from Arduino Website): *“Shields are boards that can be plugged on top of the Arduino PCB extending its capabilities”*
- So a Shiled is, basically, an application-oriented expansion board that is managed by the main board through one of its interfaces.



Example of RPi Shield (GSM/GPRS Shield)

Arduino ZigBee Shiled



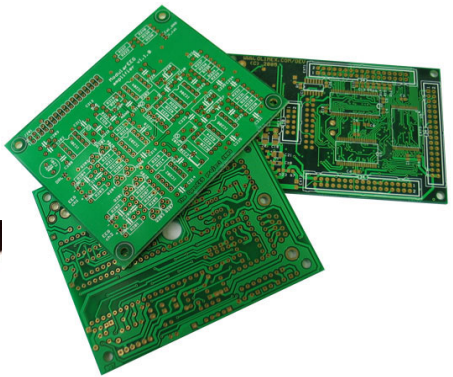
RPi to Arduino Shields Adapter

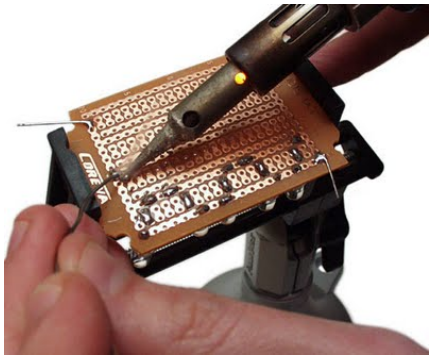
Build a Custom Shield

- Many different type of shields available on the market.
- Sometimes the range of available shields could not fit a particular application or you may have the idea of build your shield by your own.

- To build a custom shield you need:
  - An electronic CAD for design,
  - A PCB maker,
  - Shield components,
  - Assembly operations.

- Many PCB maker provide assembly service.





KiCAD EDAS

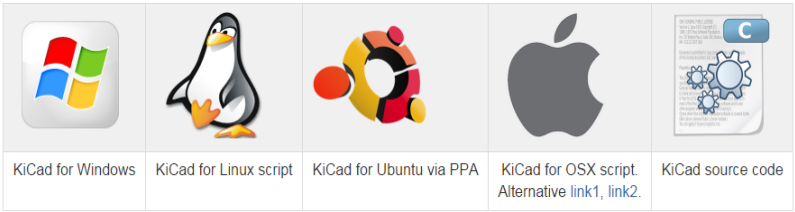
- Many different type of EDAS (Electronic Design Automation Suite) available nowadays: ORCAD, Mentor, Allegro, Altium Designer…
- Expensive Licensing, too complex for the task, too much tool inside the suite… Not suitable for RPi shields development.

- Some free open source EDAS developed through years by electronic entusiasts are now reaching a good maturity level.

- KiCAD is one of this tools!!!

Why KiCAD?

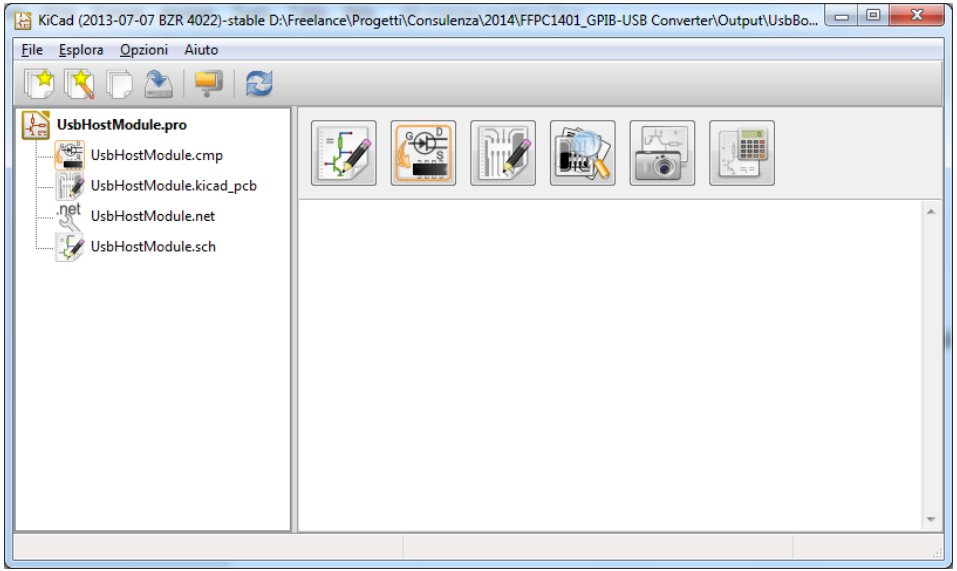
- Completely free and open source,
  - http://www.kicad-pcb.org/



- Cross-platform (Linux, MacOS and Win),
- Simple to use (getting started is only 27 pages long),
- Very large community of users that shares component libraries,
- Examples:
  - http://smisioto.no-ip.org/elettronica/kicad/kicad.htm (useful lid for hobbysts)
  - http://kicad.rohrbacher.net/quicklib.php (component generator)
  - http://www.kicadlib.org/ (libraries collection)
  - … and many others

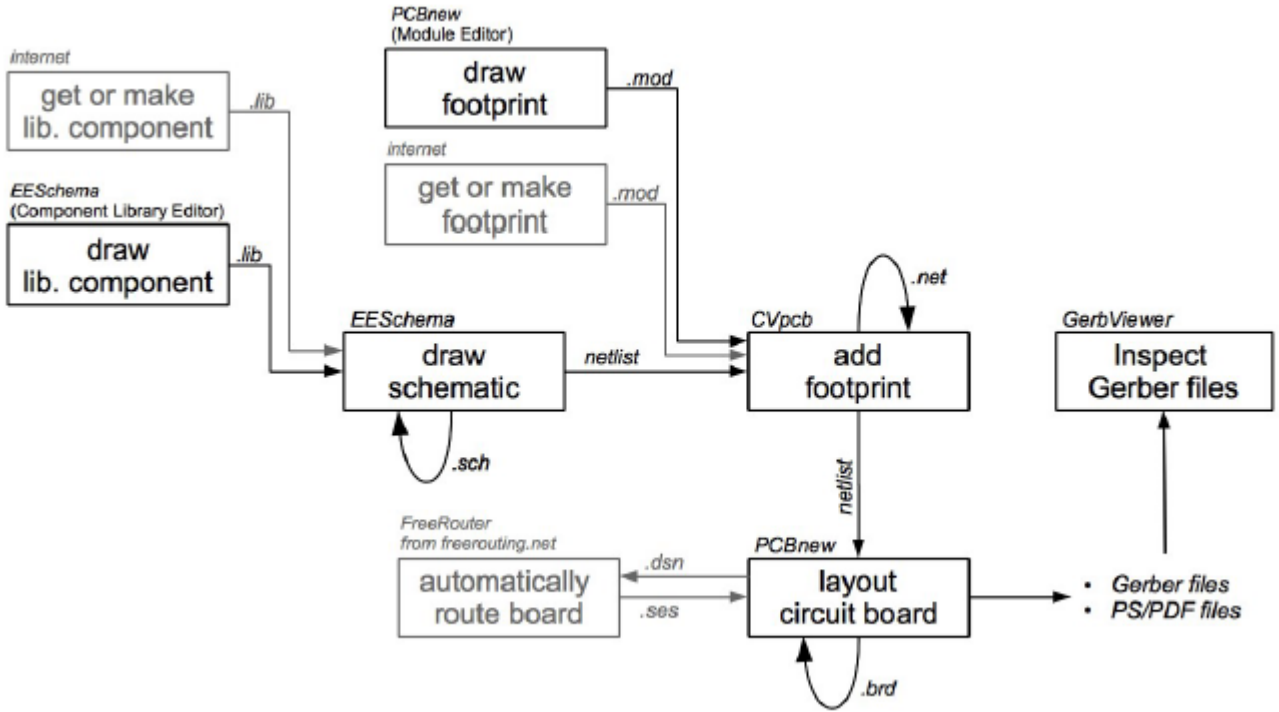
Suite Description

- It comprises a schematic editor, layout editor, gerber visualization tools and provide a simple interface for output files generation,

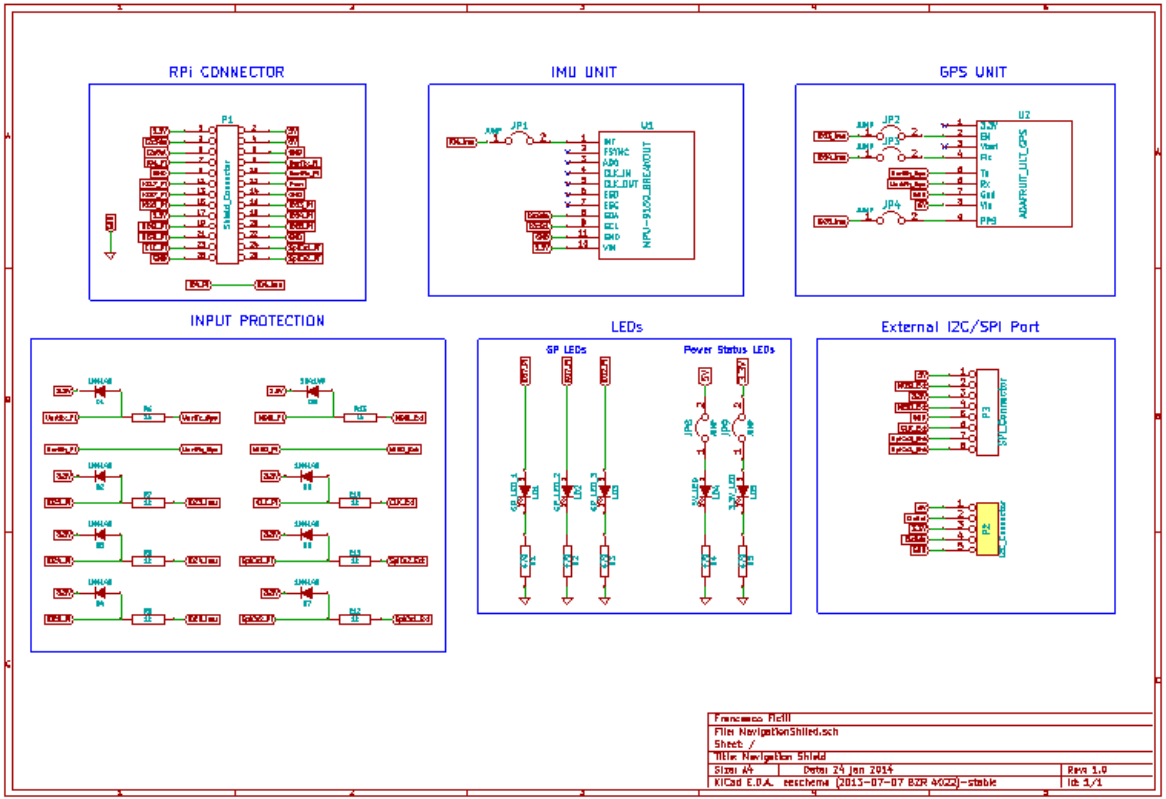


Kicad Project Manager

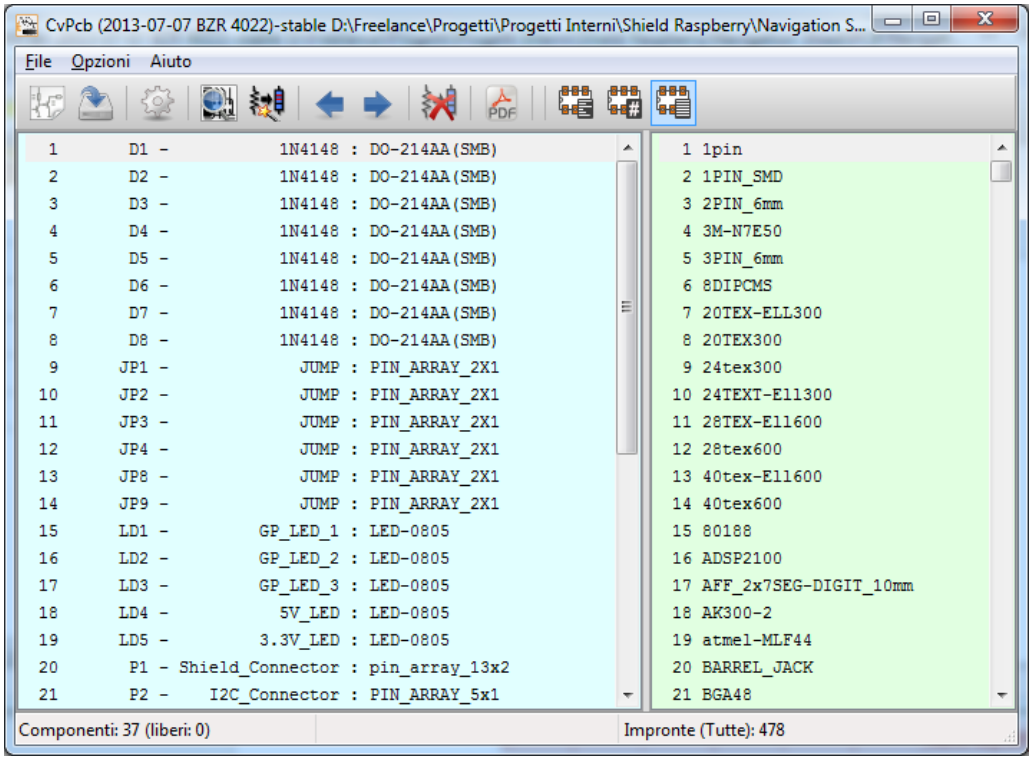
Kicad Development Process



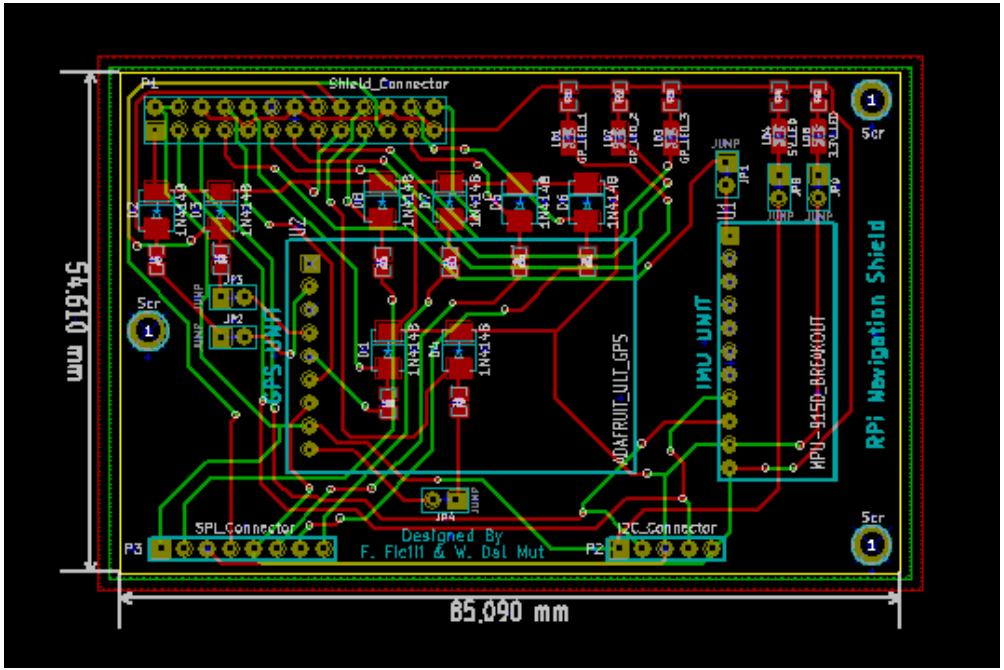
Schematic Editor



From Schematic to Layout



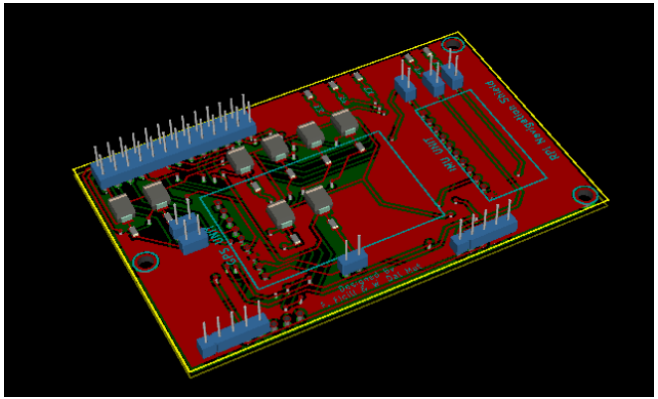
Layout Editor



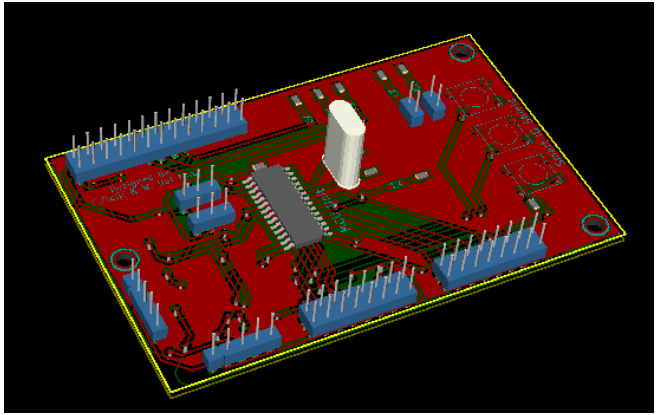
Navigation Shield Layout

Examples

- Navigation Shield
  - GPS Unit (over UART link)
  - IMU Unit (over I2C link)
  - GP LEDs
  - Possible application: rovers or drone navigation control



- Smart I/O Shield
  - 5 Analog Inputs,
  - 8 Digital I/O (2 PWM),
  - PIC18F2550 with I2C interface
  - Possible application: I/O expander for RPI (acquisition, control…).



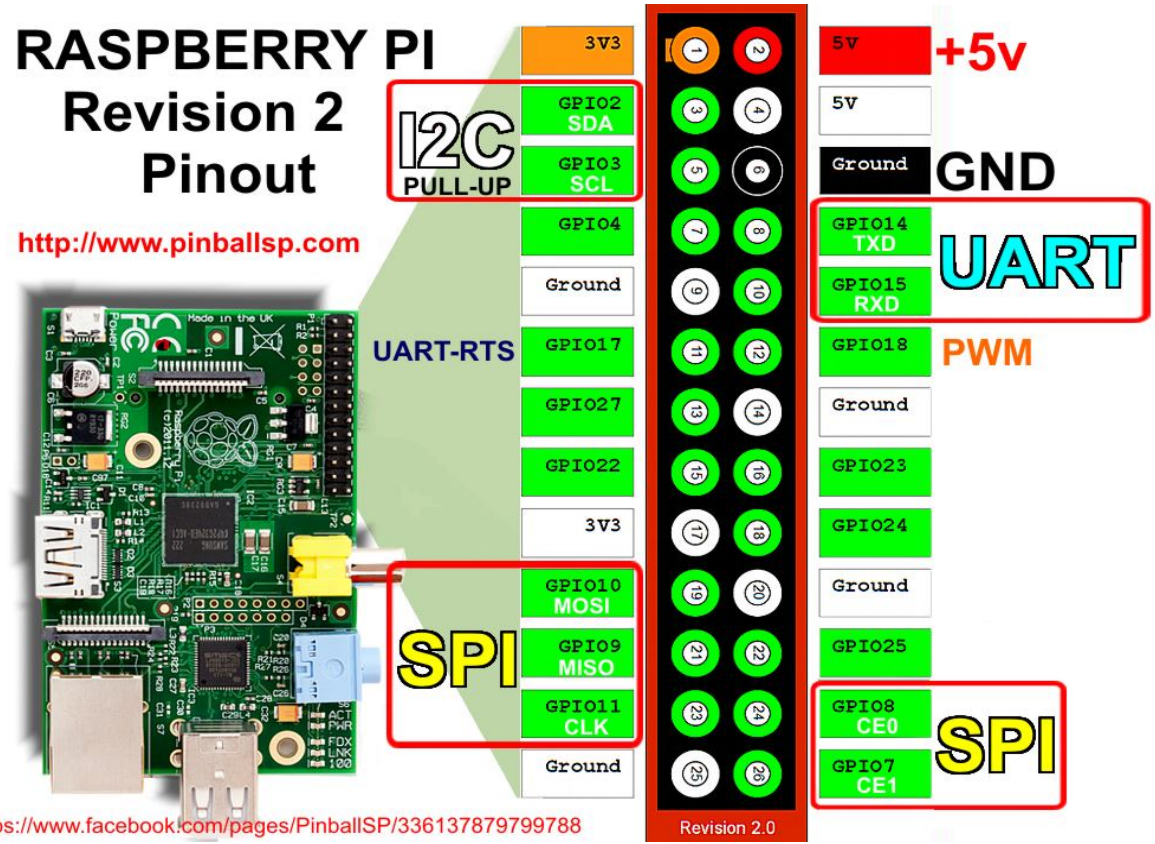
Build your prototype

- PCB (need gerber files):
  - Md srl: http://www.mdsrl.it/ (very professional service)
  - PCBProto: http://www.pcb-proto.com/ (very large pool)
  - JackAltac: http://www.jackaltac.com/ (cheaper one)

- Components (need Bill of Material):
  - Farnell, RS, Futura...

- Assembly (in case of pick-and-place need module position files)
  - Md srl is now providing an assembly service (quite expensive for small numbers)
  - ...otherwise buy a solder station

Stairway to heaven



GPIO - General Purpose In/Out

- GPIO strip provides different communication layers (board to board or on single board)
  - Generic In/Out pins (different libraries)
  - UART - Serial Communication
  - I2C - InterIntegrated Circuit
  - SPI - Serial Peripheral Interface

GPIO - Support Libraries

- Not included by default
  - http://www.airspayce.com/mikem/bcm2835/bcm2835-1.37.tar.gz

```
./configure
make
sudo make check
sudo make install
```

```
int main(void)
{
    if (!bcm2835_init()) { exit(1); }
    bcm2835_gpio_fsel(RPI_V2_GPIO_P1_11, BCM2835_GPIO_FSEL_OUTP);
    bcm2835_gpio_fsel(RPI_V2_GPIO_P1_13, BCM2835_GPIO_FSEL_OUTP);
    bcm2835_gpio_fsel(RPI_V2_GPIO_P1_15, BCM2835_GPIO_FSEL_OUTP);

    bcm2835_gpio_write(RPI_V2_GPIO_P1_11, HIGH);
    bcm2835_gpio_write(RPI_V2_GPIO_P1_13, HIGH);
    bcm2835_gpio_write(RPI_V2_GPIO_P1_15, HIGH);

    return 0;
}
```

```c
int main(void)
{
    if (!bcm2835_init()) { exit(1); }
    bcm2835_gpio_fsel(RPI_V2_GPIO_P1_11, BCM2835_GPIO_FSEL_OUTP);
    bcm2835_gpio_fsel(RPI_V2_GPIO_P1_13, BCM2835_GPIO_FSEL_OUTP);
    bcm2835_gpio_fsel(RPI_V2_GPIO_P1_15, BCM2835_GPIO_FSEL_OUTP);

    uint8_t c = 0;
    while (1) {
        bcm2835_gpio_write(RPI_V2_GPIO_P1_11, c);
        bcm2835_gpio_write(RPI_V2_GPIO_P1_13, c);
        bcm2835_gpio_write(RPI_V2_GPIO_P1_15, c);

        c = !c;
        sleep(1);
    }

    return 0;
}
```

GPIO - BCM2835 - board versions

- Model A
  - RPI_GPIO_P1_03
  - RPI_GPIO_P1_05
  - RPI_GPIO_P1_07
- Model B
  - RPI_V2_GPIO_P1_03
  - RPI_V2_GPIO_P1_05
  - RPI_V2_GPIO_P1_07
- Model B+

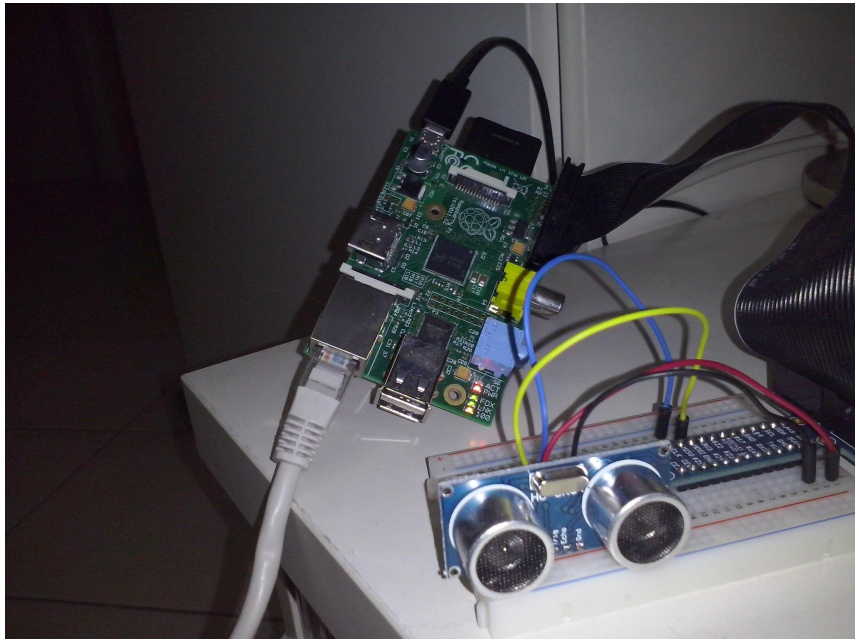
  RPI_BPLUS_GPIO_J8_03

  RPI_BPLUS_GPIO_J8_05

  RPI_BPLUS_GPIO_J8_07

http://www.airspayce.com/mikem/bcm2835/bcm2835_8h_source.html

GPIO in action



Low-cost ultrasonic sensors turn on/off a pin in order to convert real distances

HC-SR04

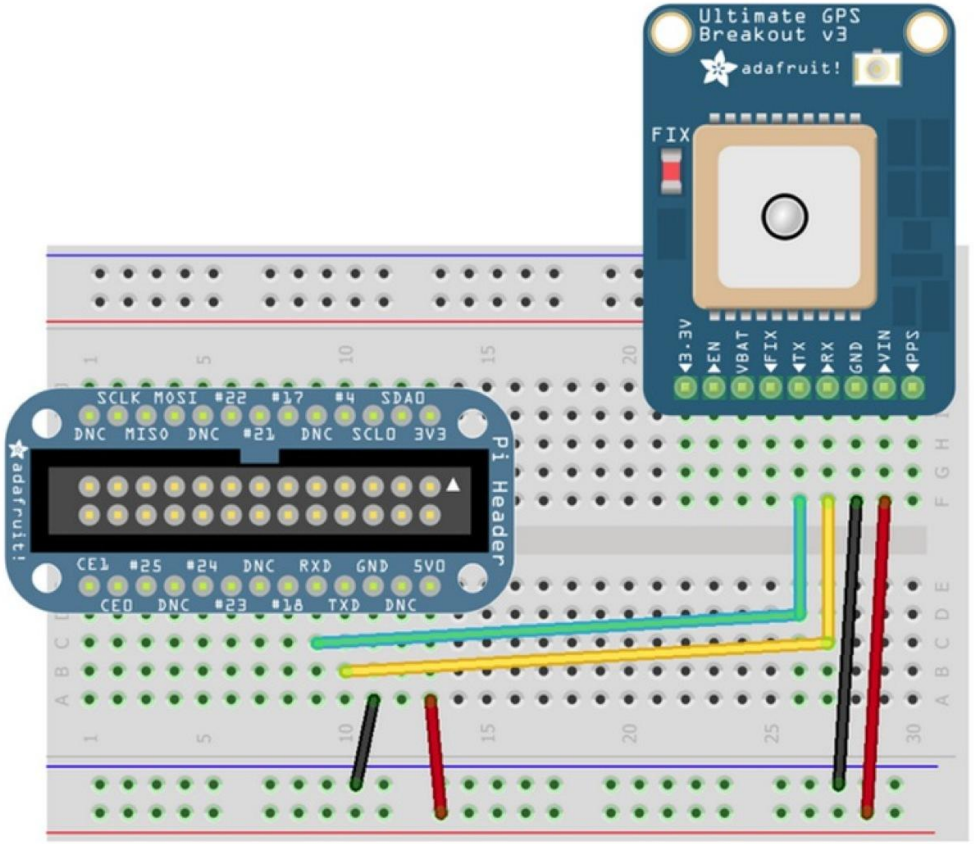
https://github.com/wdalmut/libultrasonic

UART for Raspberry Pi

- 2 functional modes
  - TTY for terminal connections (serial console)
  - For other micros and components
- Peripheral mode need few configurations
  - Appears as `/dev/ttyAMA0`
  - http://elinux.

    org/RPi_Serial_Connection#Connection_to_a_microcontroller_or_other_peripheral

Connect your Raspberry PI



Initialize your serial module

```
void serial_init(void)
{
   uart0_filestream = open(“/dev/ttyAMA0”, O_RDWR | O_NOCTTY |
O_NDELAY);

   if (uart0_filestream == -1)
   {
       //TODO error handling...
   }
}
```

Config your UART registry

```
void serial_config(void)
{
    struct termios options;
    tcgetattr(uart0_filestream, &options);
    options.c_cflag = B9600 | CS8 | CLOCAL | CREAD;
    options.c_iflag = IGNPAR;
    options.c_oflag = 0;
    options.c_lflag = 0;
    tcflush(uart0_filestream, TCIFLUSH);
    tcsetattr(uart0_filestream, TCSANOW, &options);
}
```

Send data to your peripheral

```
int count = write(uart0_filestream, data, dataLen);
```

Pay attention with strings (char vectors) because dataLen doesn’t include the termination byte (“\0”)

Read data from your peripheral

```
int rx_length = read(uart0_filestream, (void*)(&c), len);
```

A navigation shield for Raspberry Pi

Adafruit Ultimate GPS Breakout v3

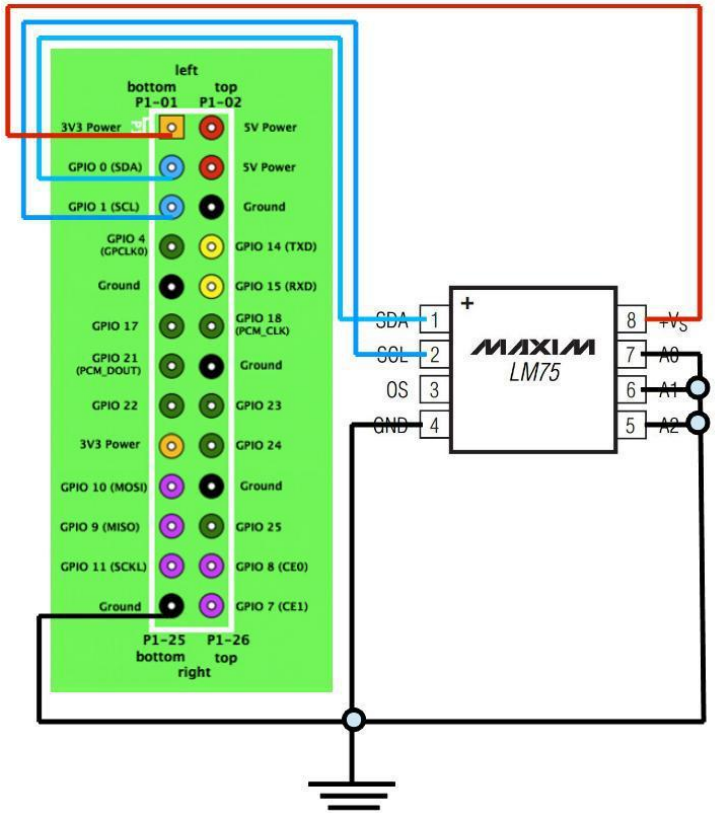
UART GPS module

https://github.com/wdalmut/libgps

I2C - Inter-Integrated Circuit

- Multiple components uses I2C in order to communicate
- Raspberry PI supports also this peripherals as a master node
- On raspberry the I2C is disabled by default, you have to enable it!
  - Enable it in your: `/etc/modprobe.d/raspi-blacklist.conf`

How to connect it?



I2C - Initialize it

```
void mma7660fc_init(void)
{
   i2cdev = open("/dev/i2c-1", O_RDWR);
   if (i2cdev == -1)
   {
      //TODO handle errors
   }
}
```

I2C - Write and control your slaves!

```
void mma7660fc_on(void)
{
   uint8_t buffer[2];

   buffer[0] = MMA7660_MODE;
   buffer[1] = MMA7660_ACTIVE;
   ioctl(i2cdev, I2C_SLAVE, MMA7660_ADDR);
   write(i2cdev, buffer, 2);
}
```

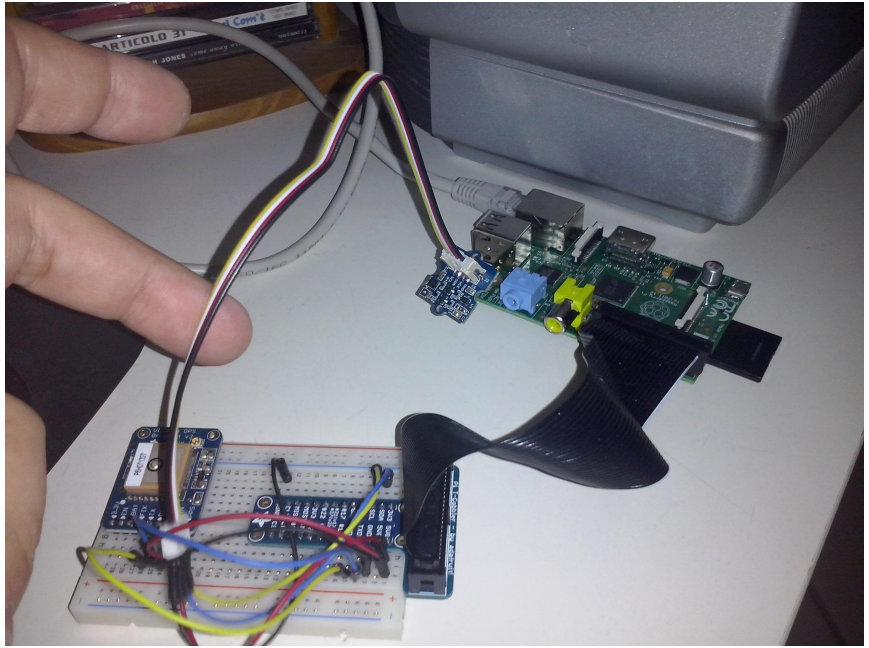
Read from your slaves

```
void mma7660fc_get(accel_t *data)
{
   uint8_t buffer[11];

   ioctl(i2cdev, I2C_SLAVE, MMA7660_ADDR);
   read(i2cdev, buffer, 3); //only x,y,z

   data->x = mma7660fc_convert(buffer[0]);
   data->y = mma7660fc_convert(buffer[1]);
   data->z = mma7660fc_convert(buffer[2]);
}
```

i2c in action



MMA7660

- 3-axis accelerometer
- Tilt detection
- Movement detection

https://github.com/wdalmut/libimu

Higher level with Golang

```
package gps
import(
   // #cgo LDFLAGS: -lgps -lm
   // #include <gps.h>
   "C"
)

func GpsLocation() (float64, float64) {
   var data C.loc_t
   var lat, lon float64

   C.gps_location(&data)

   lat = float64(data.latitude)
   lon = float64(data.longitude)

   return lat, lon
}
```

Any question?

Thanks for listening