Intel Do-It-Yourself Challenge Arduino Motor Shield

Nicolas Vailliet
www.Intel-Software-Academic-Program.com
paul.guermonprez@intel.com
Intel Software

2014-02-01

Software

We'll need

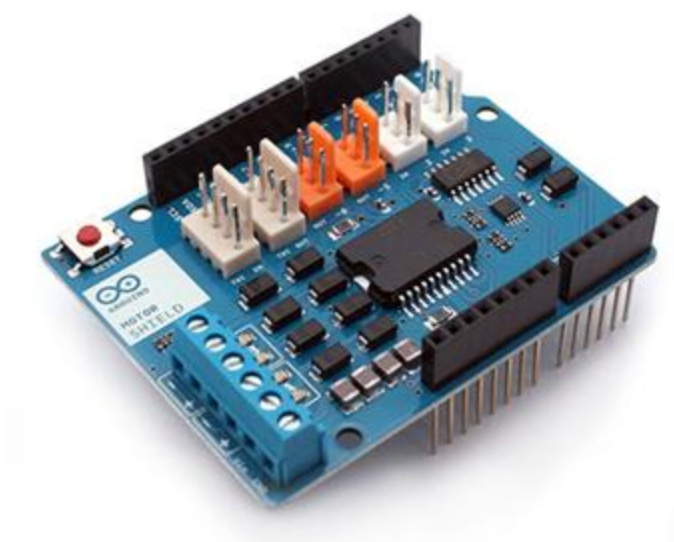
Arduino Motor Shield

Galileo board booted with full Yocto image, network and ssh.

A motor.

Arduino Motor Shield.

http://arduino.cc/en/Main/ArduinoMotorShieldR3

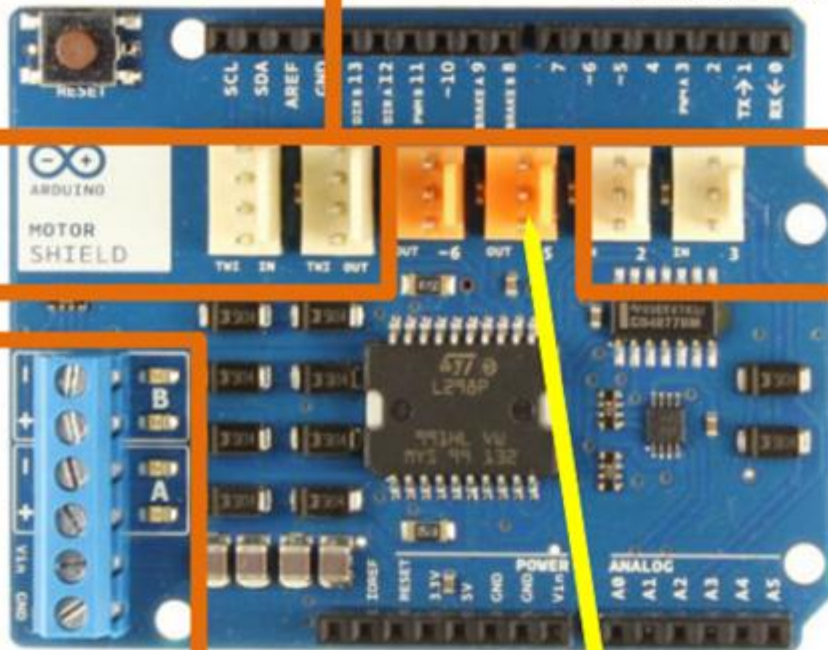


The shield

GPIO.
Eg: pins 12 & 13 control
channel A&B direction.

Connect a TWI LCD
screen here!

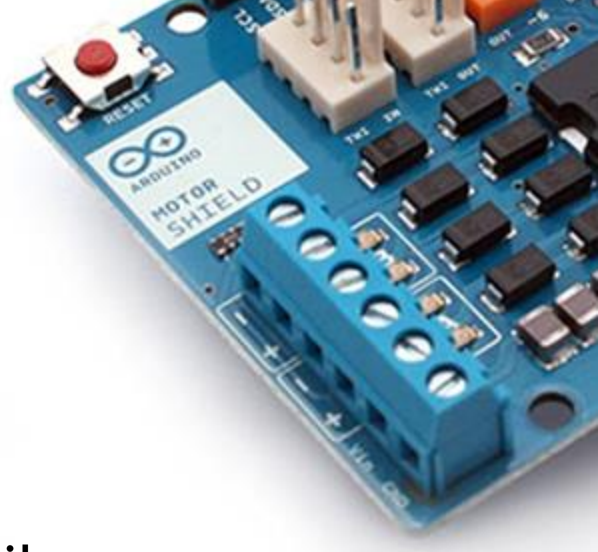
Add a sensor here.
Eg: a linear potentiometer



Plug here one or
two motors on
channel A and/or B.
Add an extra power
source if needed.

Send a 3-bit value to a
device with these pins.

Hello Motor



Let’s start

Hardware

First, be sure your Galileo is off.

Plug the Arduino Motor Shield on your Galileo.

Turn the board on, and connect with SSH.

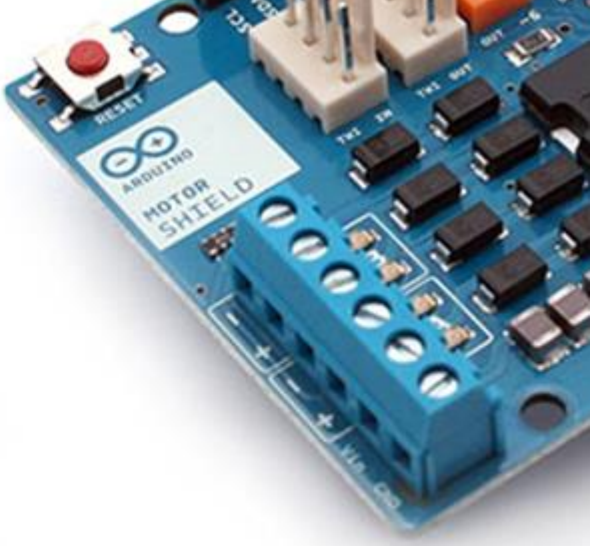
Plug a motor on channel A

Software

The motor can be controlled with GPIO, easily accessible from /sys/class/gpio/

Run the script from the next slide.

If it does not work, we’ll see why in the PWM section.



Script

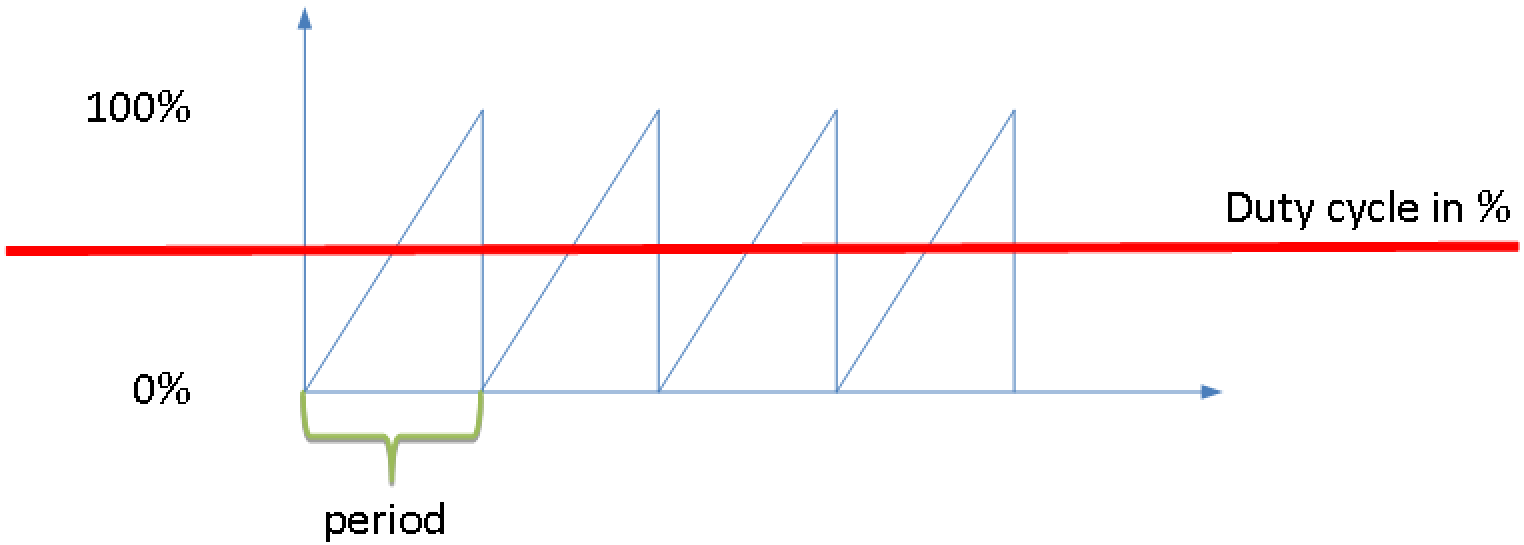
```
# disable pins 12 & 9 just in case
echo -n “12" > /sys/class/gpio/unexport
echo -n “9" > /sys/class/gpio/unexport

# enable pins 12 & 9
echo -n “12" > /sys/class/gpio/export
echo -n "9" > /sys/class/gpio/export

# we are going to send info on these pins
echo -n "out" > /sys/class/gpio/gpio9/direction
echo -n "out" > /sys/class/gpio/gpio12/direction

# brake on channel A is off
echo -n “0" > /sys/class/gpio/gpio9/value
# moving forward on channel A
echo -n “1” > /sys/class/gpio/gpio12/value
```

Pulse-Width Modulation

PWM

Pulse-Width Modulation

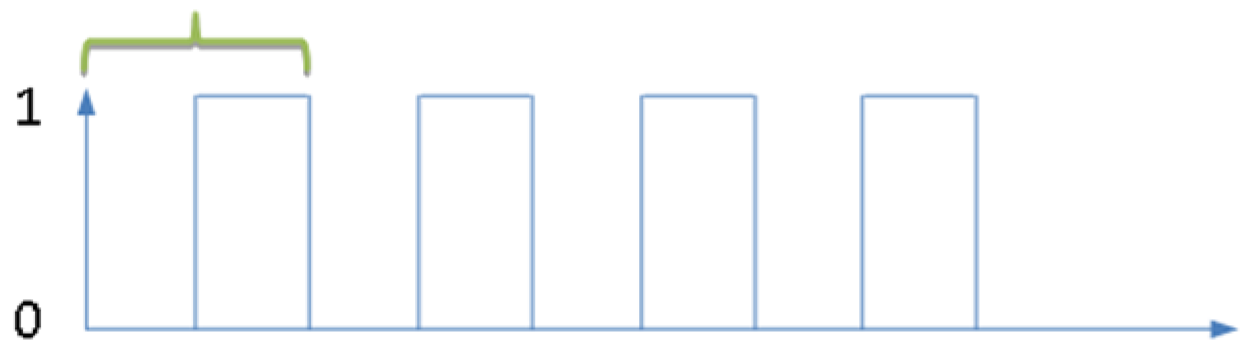
PWM stands for Pulse-Width Modulation and helps you to generate a periodic signal on a pin.



PWM

Pulse-Width Modulation

When you specify period and duty cycle values, the board generates the output HIGH only when the blue curve is over the red line. Otherwise, output is 0. In other words, you generate a squared signal which looks like this :



Warnings !

Warning !

Devices could be **damaged** by heat if you misconfigure IO ports. If it happens, don’t touch your devices !

Unplug the power supply and let it cool down 10 minutes.

Warning !

Considering channel A, you will need to set pin 12 to 0 before modifying pwm3 period and duty_cycle values.

PWM Script

```
# configuring PWM3 to control channel A voltage.
echo -n "3" > /sys/class/pwm/pwmchip0/export
# reverse command is the same but writing 3 into unexport

echo -n "1" > /sys/class/pwm/pwmchip0/pwm3/enable
# reverse command is printing 0 instead of 1


# PWM period write period in nanoseconds (here 5 milliseconds)
echo -n “5000000" > /sys/class/pwm/pwmchip0/pwm3/period

echo -n “650000" > /sys/class/pwm/pwmchip0/pwm3/duty_cycle
# PWM duty cycle write in nanoseconds.
# Simplified formula is duty cycle = ??0000
# (with ?? the percentage you want). Example with 65%
```

Summary script

Summary

Pins Channel A

(in parenthesis, values for the LSB image on SD card)

Control pins are: 3 (18), 9 (19)and 12 (38)
Brake : 9 (19)
Direction : 12 (38)
Speed : PWM3 (mapped on 3)

Pins Channel B

Control pins are: 11 (25, 8 (26) and 13 (39)
Brake : 8 (26)
Direction : 13 (39)
Speed : PWM4 (mapped on 11)

Commands

```
# print information about GPIO
cat /sys/kernel/debug/gpio
# exporting GPIO Port to file system
echo -n “9" > /sys/class/gpio/export
```

Software

Summary

```
# GPIO Port Direction (in or out)
echo -n “in" > /sys/class/gpio/gpio9/direction

# Setting GPIO Port Drive Configuration
# "pullup", "pulldown", "strong", or "hiz"
# "strong" is recommended.
echo -n "strong" > /sys/class/gpio/gpio9/drive

# Reading GPIO Port
cat /sys/class/gpio/gpio12/value
# Writing GPIO Port
echo -n "1" > /sys/class/gpio/gpio12/value
echo -n "0" > /sys/class/gpio/gpio12/value

# Reading analog value
cat /sys/bus/iio/devices/iio
# \:device0/in_voltage0_raw

# http://www.malinov.com/Home/sergey-s-blog/intelgalileo-programminggpiofromlinux
# Please refer to Sergey’s blog about IO mapping on Galileo.
```

Software

Arduino Sketch and C versions

Arduino sketch version

Of course, you can do all of this with an Arduino sketch:

```
int delay_time = 500;
void setup() {
		pinMode(12, OUTPUT);
		pinMode(9, OUTPUT);
}
void loop(){
		digitalWrite(9, LOW);
		digitalWrite(12, HIGH);
		analogWrite(3, 255);		//Full speed on channel A
		delay(delay_time );
		analogWrite(3, 50);		//Low speed on channel A
		delay(delay_time);
		digitalWrite(12,LOW);
		delay(delay_time);
}
```

C version

Use the large Yocto LSB image

If you have installed the full Linux Standard Base image, you should be able to write a C program to use your motors.

Open IO files in sysfs and do the same as previously!

With an SD card and embedded GCC, you can easily compile and run your program on the board!

C version

Compilation

If you want to cross-compile your program before loading it on the board, use Yocto cross-compiler.

The procedure is the same than to create a Linux image, except the final bitbake command that is:

bitbake image-sdk –c populate_sdk

source /opt/poky/1.4.2/environment-setup-x86_64-poky-linux

${CC} myfile.c –o myfile

.