Chapter D

Additional components - Tutorials

This appendix is the continuation of what was previously presented in part III, p.51 of this report. We will present here a series of tutorials for the components implemented in the additional components part of the framework. For each component, we will give a link to the corresponding datasheet, an estimation for the price of the component by the time we write these appendices, its type (GPIO, PWM, I²C or Analog), a description of its operating principle, a Java code example using the Raspoid framework to interact with the component and the circuit corresponding to the example, schematized with the Fritzing tool[1].

Prices are just given as an indication. There exists a large variety of similar components. It is often possible to find cheaper components on the web, and prices often decrease when the purchased quantity increases.

[1] http://fritzing.org/

D.1 LED, LED PWM & AutoFlash LED

Information

- Datasheet: http://raspoid.com/download/datasheet/LED.
- Price: ~0.20€[a].
- Type: GPIO.

[a] http://be.farnell.com/multicomp/mclf056gd/led-blinking-5mm-green/dp/1581189

D.1.1 Operating principle

LED

A LED is a Light-Emitting Diode. This is a semiconductor that turns electric energy into light energy via a PN junction. Depending on the specifications of the LED, the wavelength of the emitted light can be of any value such as infrared, visible (red, orange, yellow, green, blue, violet), or other.

It is important to not exceed the maximum current allowed through a LED (typically from 10 to 30mA for a basic low power LED) to not destroy it. It is also important to respect the polarity of the diode: we connect the "-" pole to the cathode "-" and therefore the "+" pole to the anode "+".

LED PWM

A PWM LED refers to a LED controlled with a PWM signal. With a PWM signal, it is possible to control the energy sent to the LED, and therefore the brightness intensity of the LED.

Auto-flash LED

The auto-flash LED is a LED that can randomly flash between colors when powered on.

D.1.2 Example of use with Raspoid

Example 1 - 4 simple LEDs

In this first example, we use four LEDs of different colors. Each LED is controlled from the Raspberry Pi with the GPIO pins, as presented in the code listing D.1.

```
1  public static void main(String[] args) {
2      LED white = new LED(GPIOPin.GPIO_26);
3      LED green = new LED(GPIOPin.GPIO_27);
4      LED red = new LED(GPIOPin.GPIO_28);
5      LED yellow = new LED(GPIOPin.GPIO_29);
6
7      while (true) {
8          white.toggle();
9          Tools.sleepMilliseconds(250);
10         green.toggle();
11         Tools.sleepMilliseconds(250);
12         red.toggle();
13         Tools.sleepMilliseconds(250);
14         yellow.toggle();
15         Tools.sleepMilliseconds(250);
16     }
17 }
```

Listing D.1: 4 simple LEDs - program example with Raspoid.

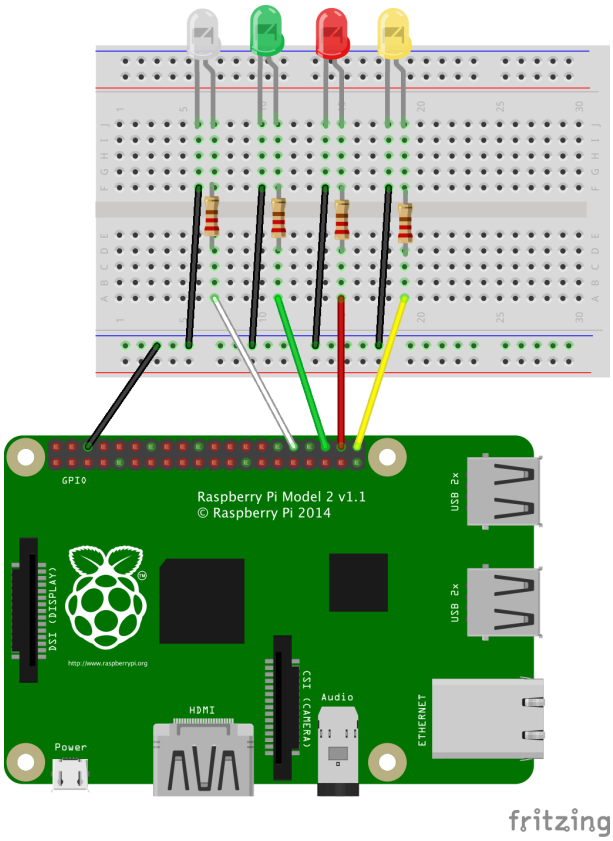

Figure D.1: Example of 4 simple LEDs controlled with ON/OFF signals from the GPIO - circuit.

Example 2 - LED PWM

In this second example, a LED is controlled with a PWM signal, generated from a PCA9685 board. The PCA9685 is controlled from the Raspberry Pi with the I²C protocol. In the code listing D.2, the LED blinks with a dimmer effect.

```
public static void main(String[] args) {
    LEDPWM led = new LEDPWM(new PCA9685(), PCA9685Channel.CHANNEL_00);

    while(true) {
        for(int i=0; i<100; i++) {
            led.setIntensity(i);
            Tools.sleepMilliseconds(5);
        }

        for(int i = 100; i > 0; i--) {
            led.setIntensity(i);
            Tools.sleepMilliseconds(5);
        }
    }
}
```

Listing D.2: Example of a LED controlled with a PWM signal - program.

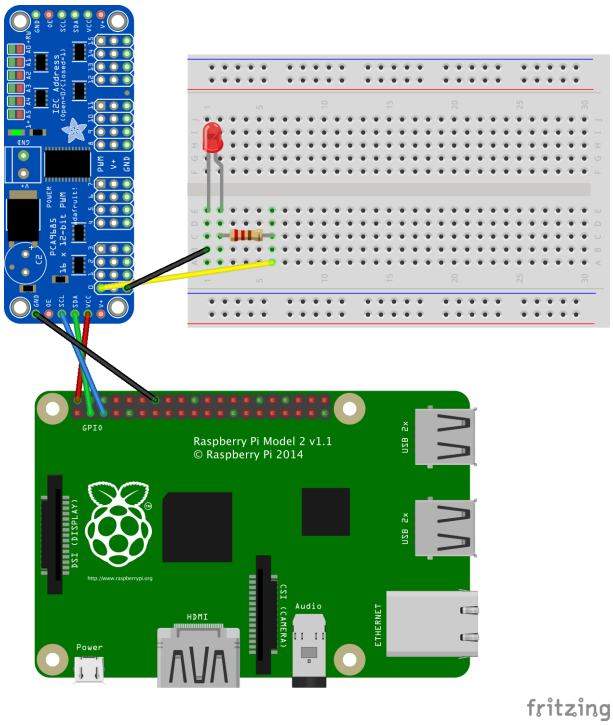

Figure D.2: Example of a LED controlled with a PWM signal - circuit.

D.2 Button & touch switch

Information

- Datasheet: http://raspoid.com/download/datasheet/Button.
- Price: ~0,50€[a].
- Type: GPIO.

[a] https://www.sparkfun.com/products/9190

D.2.1 Operating principle

Button

A button is a switch which lets the current flow or not. It can be in a direct (the current flows when the button is pressed) or reverse mode (the current flows when the button is released).

Touch switch

A touch switch is a button which state is toggled at each finger touch. First touch: the signal goes high; second touch: the signal goes down; and so on.

D.2.2 Example of use with Raspoid

In this example, a simple button is connected to the Raspberry Pi. When the button is pressed, an interruption is detected by the Raspberry Pi, and an event is retrieved by the Raspoid program, as presented in the code listing D.3. The same process happens when the button is released.

```
1 public static void main(String[] args) {
2     Button button = new Button(GPIOPin.GPIO_00);
3
4     // Add a listener for button pressed/released events
5     button.getGpioPinDigitalInput().addListener((GpioPinListenerDigital)
6             (GpioPinDigitalStateChangeEvent event) -> Tools.log(button.isPressed() ?
      "button pressed" : "button released"));
7
8     Tools.sleepMilliseconds(15000);
9 }
```

Listing D.3: Example of a simple button connected to a GPIO pin - program.

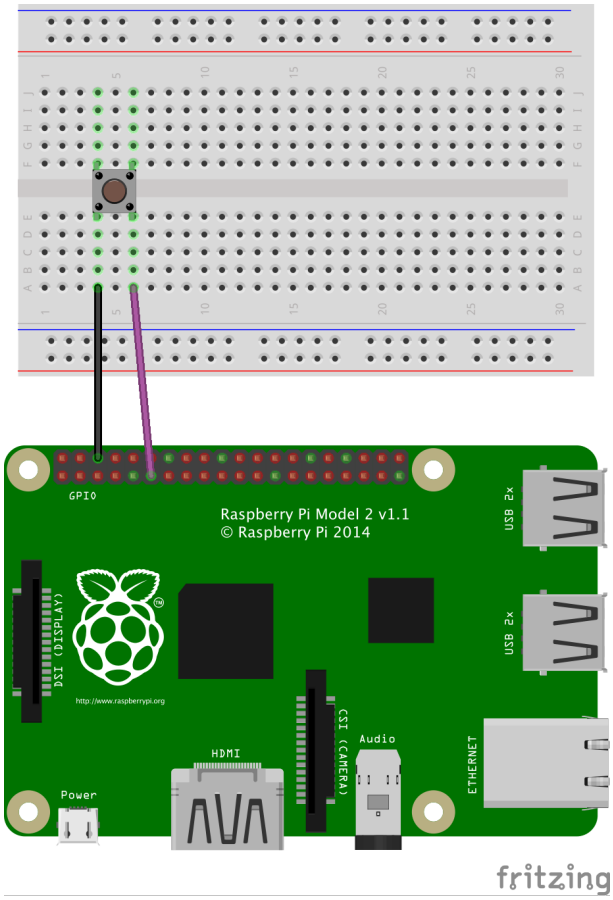

Figure D.3: Example of a simple button connected to a GPIO pin. Black: ground; Violet: GPIO 0.

D.3 LCD Display - LCM1602

Information

- Datasheet: http://raspoid.com/download/datasheet/LCM_MODULE.
- Price: ~6.77€ (~1.67€[a] for the I²C serial interface module + ~5.10€[b] for the LCD display).
- Type: I²C.

[a] http://www.amazon.fr/dp/B00GBSWOWW
[b] http://www.amazon.fr/dp/B009GEPZRE

D.3.1 Operating principle

The LCM1602 is an LCD display of 2 lines and 16 columns (it can display 32 ASCII characters), that can directly be used with the I²C protocol. We implemented a complete wrapper for this component in the Raspoid framework so that it can easily be used.

D.3.2 Example of use with Raspoid

In this example, the LCD1602 is connected to the LCM1602 (generally, the two components are already welded when purchased). The LCM1602 is connected to the ground, to the 5v power, and to the SDA and SCL pins of the Raspberry Pi.
As presented in the code listing D.4, the LCD displays the current time in the "HH:mm:ss" format, and updates it 4 times per second to avoid delays between the displaying and real time.

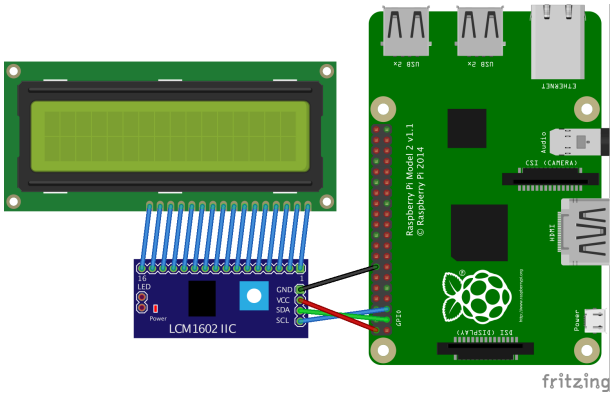

Figure D.4: Example of use of an LCM1602 - circuit.

```java
public static void main(String[] args) {
    LCM1602 display = new LCM1602();
    display.setDisplay(true, false, false);

    // To execute on exit (clean screen)
    Runtime.getRuntime().addShutdownHook(new Thread(() ->
            display.disableDisplay()));

    display.writeText(0, 0, "Raspoid  Welcome");

    SimpleDateFormat formatter = new SimpleDateFormat("HH:mm:ss");
    while (true) {
        display.writeText(4, 1, formatter.format(new Date()));
        Tools.sleepMilliseconds(250);
    }
}
```

Listing D.4: Example of use of an LCM1602 - program.

D.4 Joystick

Information

- Datasheet: http://raspoid.com/download/datasheet/Joystick.
- Price: ~2.32€[a].
- Type: Analog.

[a] http://www.amazon.fr/dp/B01AXOG1FK

D.4.1 Operating principle

A joystick is mainly composed of two analog outputs. The intensity of the output signal is directly related to the position of the joystick, in the x and y axis (the two analog outputs). From the ADC, we can then retrieve (x,y) coordinates with x and y varying in a range from 0 to 255: (0,0)=bottom left; (126,126)=default position; (255,255)=top right; etc.

The joystick we experimented with with here is an "analog" joystick very similar to those on PlayStation 2 controllers. Positions in x and y are translated from potentiometers (one for each axis). This kind of joystick also acts has a press button. When the button is pressed, the corresponding pin is linked to the ground, and an interruption can be detected.

D.4.2 Example of use with Raspoid

In this example, a joystick is connected to a PCF8591 (ADC). This ADC is connected to the Raspberry Pi and the I²C protocol is used to retrieve digital values converted from the joystick, and stored on the registers of the PCF8591.

```
1 public static void main(String[] args) {
2     Joystick joystick = new Joystick(new PCF8591(), PCF8591InputChannel.CHANNEL_0,
      PCF8591InputChannel.CHANNEL_1, PCF8591InputChannel.CHANNEL_2);
3
4     while(true) {
5         Tools.log(joystick.getPosition()); // press-down, up/down, left/right, home
6         Tools.sleepMilliseconds(250);
7     }
8 }
```

Listing D.5: Example of use of a joystick - program.

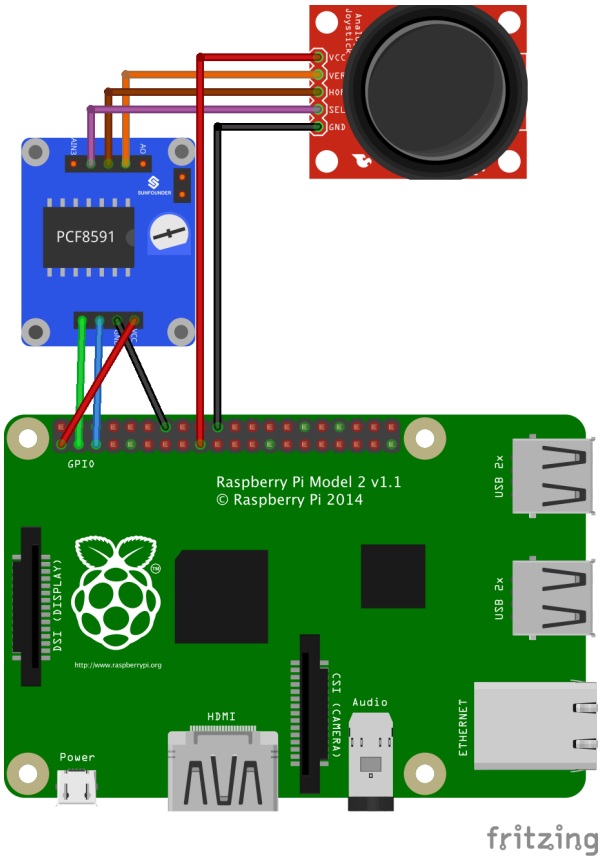

Figure D.5: Example of use of a joystick - circuit.

D.5 Rotary encoder

Information

- Datasheet: http://raspoid.com/download/datasheet/rotaryEncoder.
- Price: ~1.83€[a].
- Type: GPIO.

[a] http://be.farnell.com/fr-BE/bourns/pel12s-4224s-n1024/rotary-encoder-incremental-24/dp/1857564

D.5.1 Operating principle

A rotary encoder is used here to detect the orientation of a shaft. Two digital input pins are used by this module. The signals on those two pins are analyzed and combined to deduce the direction of new rotations. We then use a counter to represent the orientation: when turned to the right, the counter is incremented; when turned to the left, it is decremented.

D.5.2 Example of use with Raspoid

In the following example, a rotary encoder is connected to the Raspberry Pi by using three GPIO pins. Digital values read from those pins are combined to determine the direction of new rotations. Internally, a counter is used to represent the position of the shaft of the rotary encoder. When turning to the right or to the left, the updated value of the counter will be printed in the standard output.

```
1  public static void main(String[] args) {
2      RotaryEncoder rotaryEncoder = new RotaryEncoder(GPIOPin.GPIO_27, GPIOPin.GPIO_28
       , GPIOPin.GPIO_29);
3
4      rotaryEncoder.getGpioPinDigitalInput().addListener((GpioPinListenerDigital)
5              (GpioPinDigitalStateChangeEvent event) -> Tools.log(rotaryEncoder.
       isPressed() ? "rotary pressed" : "rotary released"));
6
7      int previousCounterValue = rotaryEncoder.getCounterValue();
8      while(true) {
9          rotaryEncoder.getEncoderTurn();
10         if(previousCounterValue != rotaryEncoder.getCounterValue()) {
11             Tools.log(rotaryEncoder.getCounterValue());
12             previousCounterValue = rotaryEncoder.getCounterValue();
13         }
14     }
15 }
```

Listing D.6: Example of use of a rotary encoder - program.

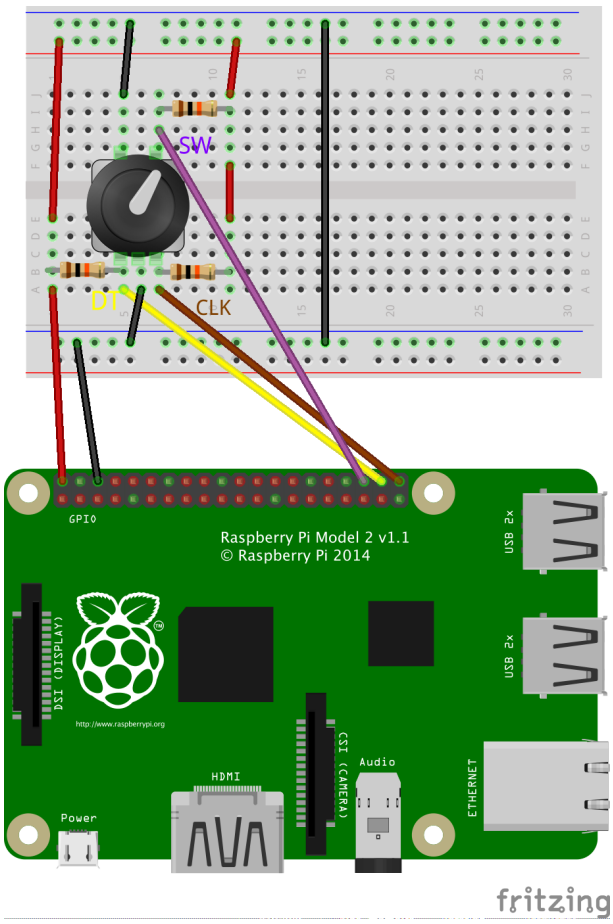

Figure D.6: Example of use of a rotary encoder - circuit.

D.6 Thermistor

Information

- Datasheet: http://raspoid.com/download/datasheet/ThermistorNTC.
- Price: ~1.20€[a].
- Type: Analog.

[a] http://be.farnell.com/fr-BE/vishay/ntcle203e3103fb0/thermistance-ctn/dp/1187039

D.6.1 Operating principle

A thermistor is made of semiconductor materials, for which the resistance varies significantly with ambient temperature. It can mainly be used to detect variations of the ambient temperature.

D.6.2 Example of use with Raspoid

In this example, a thermistor is connected to a PCF8591 (ADC). This ADC is connected to the Raspberry Pi and the I²C protocol is used to read values converted from the thermistor and stored in the registers of the PCF8591.

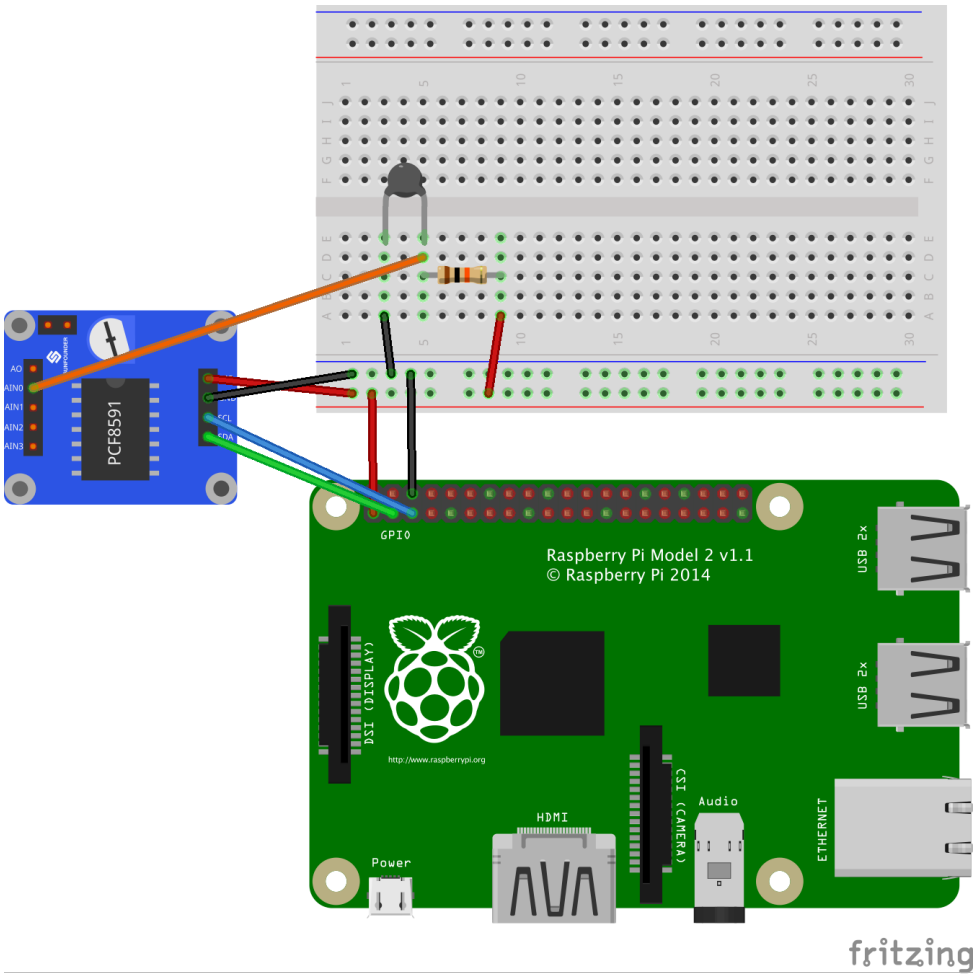

Figure D.7: Example of use of a thermistor - circuit.

```
public static void main(String[] args) {
    Thermistor thermistor = new ThermistorNTCLE203E3103SB0(new PCF8591(),
    PCF8591InputChannel.CHANNEL_0);

    while(true) {
        Tools.log(thermistor.getTemperature());
        Tools.sleepMilliseconds(500);
    }
}
```

Listing D.7: Example of use of thermistor - program.

D.7 Barometer - BMP180

Information

- Datasheet: http://raspoid.com/download/datasheet/BMP180.
- Price: ~3.00€[a].
- Type: I²C.

[a] http://www.amazon.fr/dp/B00QAG7PBU

D.7.1 Operating principle

The barometer BMP180 is a digital barometric pressure sensor. It can be used to measure air pressure and temperature. The pressure varying with the altitude, this sensor can also be used as an altimeter.

D.7.2 Example of use with Raspoid

In this example, a BMP180 is connected to the Raspberry Pi and the I²C protocol is used to communicate with the sensor and retrieve data from the registers of the component.

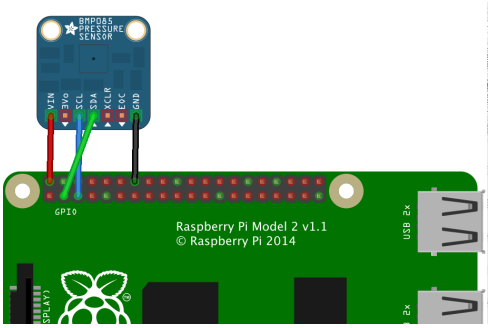

Figure D.8: Example of use of a BMP180 - circuit.

```
public static void main(String[] args) {
    BarometerBMP180 barometer = new BarometerBMP180();

    while(true) {
        Tools.log(
            "Air pressure: " + barometer.readUncompensatedPressure() + " " +
            "True air pressure: " + barometer.calculateTruePressure() + " Pa\n" +
            "Temperature: " + barometer.readUncompensatedTemperature() + " " +
            "True temperature: " + barometer.calculateTrueTemperature() + "°C\n" +
            "Altitude: " + barometer.calculateAbsoluteAltitude() + "m",
            Tools.Color.ANSI_GREEN);
        Tools.sleepMilliseconds(1000);
    }
}
```

Listing D.8: Example of use of a BMP180 - program.

D.8 IR receiver & IR transmitter

Information

Transmitter

- Datasheet: http://raspoid.com/download/datasheet/ir_led.
- Price: ~0.276€[a].
- Type: PWM.

Receiver

- Datasheet: http://raspoid.com/download/datasheet/IRReceiverOS1838B.
- Price: ~1.40€[b].
- Type: GPIO.

[a] http://be.farnell.com/fr-BE/vishay/tsal6100/ir-emitter-940nm-t-1-3-4-through/dp/1328299
[b] http://be.farnell.com/fr-BE/vishay/tsdp34138/ir-receiver-35m-940nm-38-4khz/dp/2442775

D.8.1 Operating principle

Our implementation of the infrared transmitter/receiver allows the user to easily control its robot with infrared signals.

IR Transmitter

An infrared transmitter is an infrared LED that should be connected to a Raspberry Pi's PWM pin (the frequency from the PCA9685 is not enough for infrared signals). Infrared signals can then be sent and detected with the receiver presented here after.

IR Receiver

The infrared receiver has been designed here to detect infrared signals sent from a media remote (for instance). The implementation of an infrared media remote using an infrared-LED can be made with the IR transmitter previously presented.

Figure D.9: Basic Media Remote.

D.8.2 Example of use with Raspoid

In the first part of this example, when an infrared signal is detected by the IR receiver, if the signal is correctly decoded, the result is printed in the standard output. In the second part of this example, an infrared signal is sent through the infrared LED. This signal is the same as the signal sent from the infrared media remote sold by Sunfounder and presented in figure D.9, when button 1 is pressed.

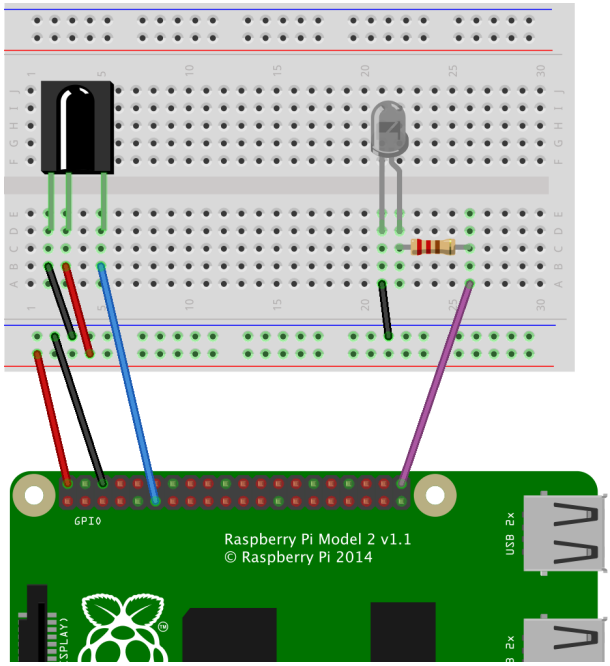

Figure D.10: Example of use of an IR receiver (on the left) and an IR transmitter (on the right) - circuit.

```
public static void main(String[] args) {
    IRReceiverOS1838B irReceiver = new IRReceiverOS1838B(GPIOPin.GPIO_00);

    while(true) {
        IRSignal newSignal = irReceiver.detectSignal();

        IRSignal signalDecoded = irReceiver.decodeIRSignal(new
    IRProtocolSunfounderMediaRemote(), newSignal);
        if(signalDecoded != null)
            Tools.log("New signal received and decoded: " + signalDecoded.getName())
    ;
        else
            Tools.log("New signal received but NOT decoded: " + newSignal);
    }
}
```

Listing D.9: Example of use of an infrared receiver - program.

```
public static void main(String[] args) {
    IRTransmitter transmitter = new IRTransmitter(PWMPin.PWM0);
    transmitter.transmitSignal(IRProtocolSunfounderMediaRemote.button1);
    Tools.log("IR signal sent");
}
```

Listing D.10: Example of use of an infrared transmitter - program.

D.9 Accelerometer - ADXL345

Information

- Datasheet: http://raspoid.com/download/datasheet/ADXL345.
- Price: ~3.99€[a].
- Type: I²C.

[a] http://www.amazon.fr/dp/B00LO306GA

D.9.1 Operating principle

The accelerometer ADXL345 is a cheap accelerometer that can be used to retrieve accelerations for x, y and z axes.

D.9.2 Example of use with Raspoid

In this example, we use the I²C protocol to control the accelerometer, and to retrieve values corresponding to (x, y, z) coordinates of the last measured acceleration vector.

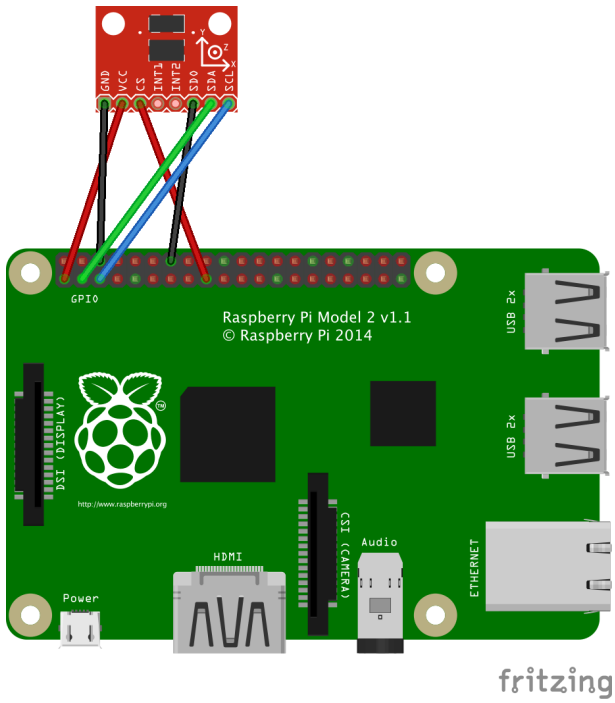

Figure D.11: Example of use of an ADXL345 accelerometer - circuit.

```
public static void main(String[] args) {
    AccelerometerADXL345 accel = new AccelerometerADXL345();

    while(true) {
        Tools.log("Acceleration: " + accel.getGAcceleration()); // {x, y, z} vector
        Tools.log("Pitch angle: " + accel.getPitchAngle() + "°");
        Tools.sleepMilliseconds(250);
    }
}
```

Listing D.11: Example of use of an ADXL345 accelerometer - program.

D.10 Sound sensor - LM358

Information

- Datasheet: http://raspoid.com/download/datasheet/SoundSensor.
- Price: ~5.99€[a].
- Type: Analog.

[a] http://www.amazon.fr/dp/B013QR8ZT6

D.10.1 Operating principle

A sound sensor is composed of a simple microphone and is able to detect ambient sound intensity by converting audio signals to analog electrical signals.

D.10.2 Example of use with Raspoid

In this example, the sound sensor is used with a PCF8591 to convert the analog signal to a digital one. The PCF8591 is connected to the Raspberry Pi and the I²C protocol is used to retrieve those digital values.

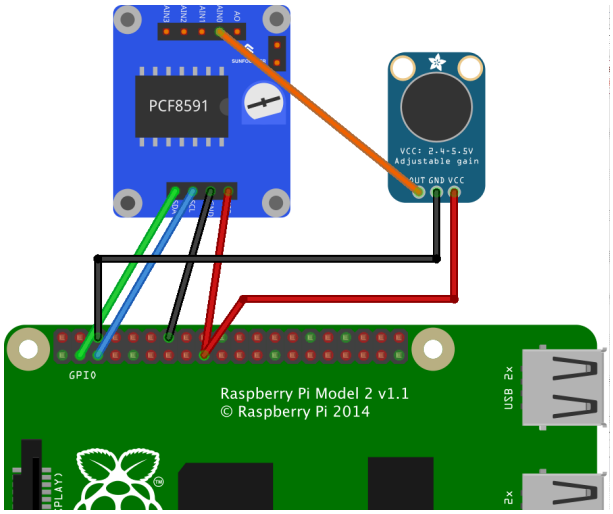

Figure D.12: Example of use of a sound sensor - circuit.

```
1 public static void main(String[] args) {
2     SoundSensor soundSensor = new SoundSensor(new PCF8591(), PCF8591InputChannel.
      CHANNEL_0);
3
4     while(true) {
5         Tools.log(soundSensor.getIntensity());
6         Tools.sleepMilliseconds(25);
7     }
8 }
```

Listing D.12: Example of use of a sound sensor - program.

D.11 Buzzer (active - passive)

Information

Active

- Datasheet: http://raspoid.com/download/datasheet/ActiveBuzzer.
- Price: ~2.10€[a].
- Type: GPIO.

Passive

- Datasheet: http://raspoid.com/download/datasheet/PassiveBuzzer.
- Price: ~1.72€[b].
- Type: PWM.

[a] http://be.farnell.com/fr-BE/pro-signal/abi-042-rc/indicator-audio/dp/1827949
[b] http://www.digikey.com/product-detail/en/tdk-corporation/SD1209T3-A1/445-2530-ND/935935

D.11.1 Operating principle

Active

An active buzzer has a built-in oscillating source. It will produce a sound at a specific frequency, as long as it is electrified.

Passive

A passive buzzer can be used to control the frequency of the generated sounds. Our implementation allows the user to play music by specifying base tones, octaves and notes duration.

D.11.2 Example of use with Raspoid

In this example, we show the use of a buzzer. This buzzer can be active or passive. The only difference in the circuit is the pin used on the Raspberry Pi. This pin must be a PWM pin for a passive buzzer (to play specific tones by adapting frequencies of the PWM signal), while it must be a classical GPIO pin for an active buzzer (it produces sound only when electrified).

In the first code listing D.13, the active buzzer will loop between beep and silence every 500 milliseconds. In the second code listing D.14, the passive buzzer will play each note of the scale "do re mi fa sol la si", and loop through octaves from 0 to 7 (500 milliseconds per tone).

```
1 public static void main(String[] args) {
2     ActiveBuzzer buzzer = new ActiveBuzzer(GPIOPin.GPIO_00, true);
3     while(true)
4         buzzer.beep(500);
5 }
```

Listing D.13: Example of use of an active buzzer - program.

```
1 public static void main(String[] args) {
2     int timePerNote = 500; // ms
3     PassiveBuzzer buzzer = new PassiveBuzzer(PWMPin.PWM1);
4
5     for(int i=0; i <= 7; i++) {
6         buzzer.playNote(BaseNote.DO_0, i, timePerNote);
7         buzzer.playNote(BaseNote.RE_0, i, timePerNote);
```

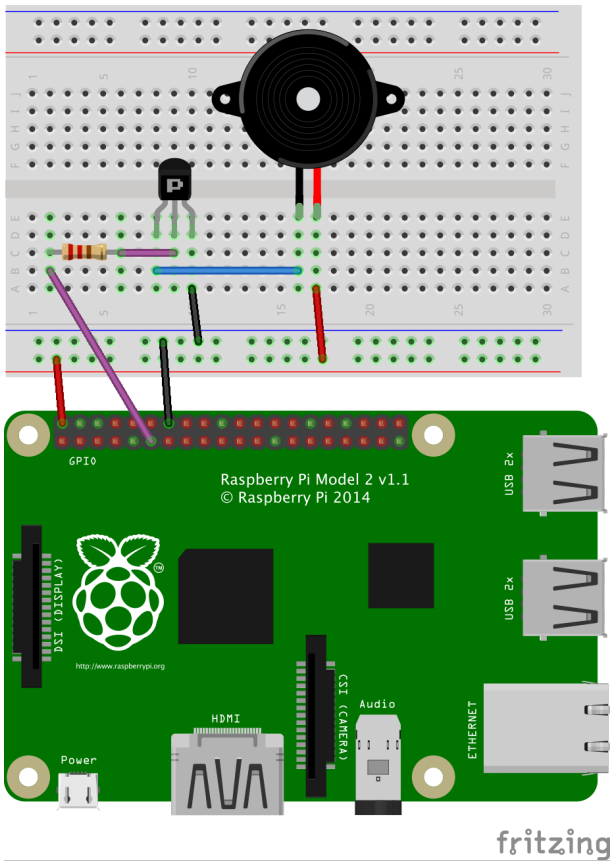

Figure D.13: Example of use of a buzzer (active or passive) - circuit.

```
8             buzzer.playNote(BaseNote.MI_0, i, timePerNote);
9             buzzer.playNote(BaseNote.FA_0, i, timePerNote);
10            buzzer.playNote(BaseNote.SOL_0, i, timePerNote);
11            buzzer.playNote(BaseNote.LA_0, i, timePerNote);
12            buzzer.playNote(BaseNote.SI_0, i, timePerNote);
13        }
14    }
```

Listing D.14: Example of use of a passive buzzer - program.

D.12 Tracking sensor - TRT5000

Information

- Datasheet: http://raspoid.com/download/datasheet/tcrt5000.
- Price: ~0.82€[a] (~7.02€[b] for a complete module).
- Type: GPIO.

[a] http://be.farnell.com/fr-BE/vishay/tcrt5000l/capteur-optique-reflectif/dp/1703519
[b] http://www.sunfounder.com/tracking-sensor-module.html

D.12.1 Operating principle

A tracking sensor uses the same principle as the infrared obstacle avoidance module. An infrared signal is emitted, and the reflected signal is received by the detector part of the module. The particularity used here, is that black color will not reflect infrared signals. Thus, the sensor can be oriented towards the ground, and used to track a black line.

D.12.2 Example of use with Raspoid

In this example, when the detected color switches between white and black, an event is triggered and the new detected color is printed in the standard output.

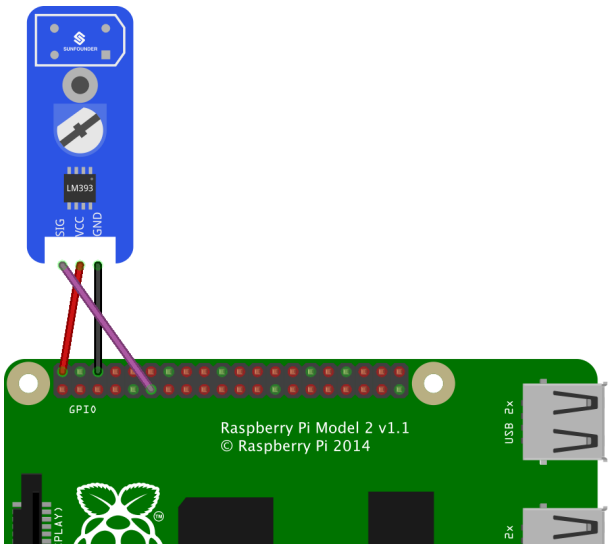

Figure D.14: Example of use of a tracking sensor - circuit.

```
public static void main(String[] args) {
    TrackingSensor trackingSensor = new TrackingSensor(GPIOPin.GPIO_00);

    trackingSensor.getGpioPinDigitalInput().addListener((GpioPinListenerDigital)
            (GpioPinDigitalStateChangeEvent event) ->
    Tools.log("Color: " + (event.getState().isLow() ? "white" : "black")));
}
```

Listing D.15: Example of use of a tracking sensor - program.

D.13 IR obstacle avoidance module

Information

- Datasheet: http://raspoid.com/download/datasheet/DifferentialComparator.
- Price: ~8.09€[a] for a complete module.
- Type: GPIO.

[a] http://www.sunfounder.com/obstacle-avoidance-sensor-module.html

D.13.1 Operating principle

The infrared obstacle avoidance module is a little chip composed of an infrared emitter and an infrared receiver. It uses the infrared reflection principle to detect obstacles: when there is no object ahead of the emitter, the receiver cannot detect signals; when there is an obstacle, the infrared light is reflected and is then detected by the receiver (it detects obstacles in a 0-7 cm range). The module presented here is sold by Sounfounder.

D.13.2 Example of use with Raspoid

In this example, an event is triggered when an obstacle is detected, or when the obstacle has disappeared. A message is simply printed in the standard output each time the status changes (obstacle detected >< no obstacle).

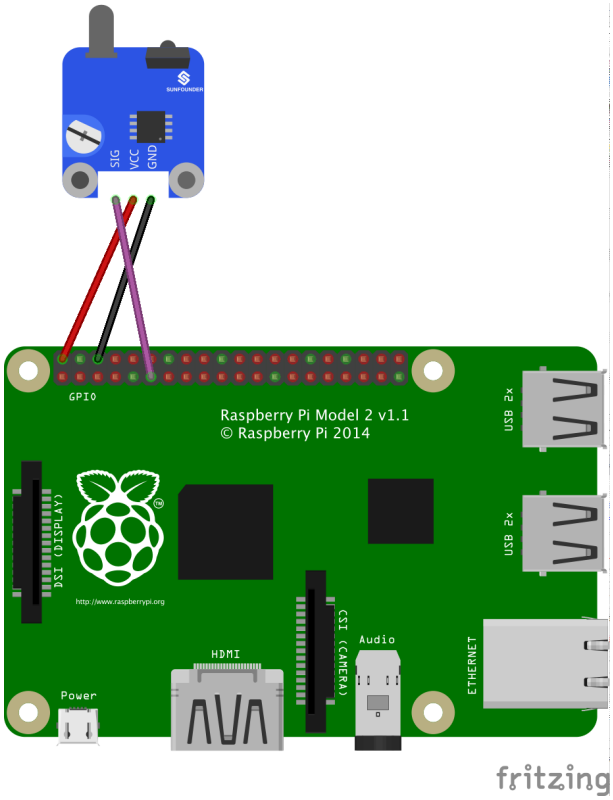

Figure D.15: Example of use of an obstacle avoidance module - circuit.

```
public static void main(String[] args) {
    IRObstacleAvoidanceModule obstacleAvoidanceModule = new
    IRObstacleAvoidanceModule(GPIOPin.GPIO_00);

    obstacleAvoidanceModule.getGpioPinDigitalInput().addListener((
    GpioPinListenerDigital)
            (GpioPinDigitalStateChangeEvent event) -> Tools.log(event.getState().
    isLow() ? "obstacle detected" : "no obstacle"));

    Tools.sleepMilliseconds(15000);
}
```

Listing D.16: Example of use of an obstacle avoidance module - program.