Laboratory: Sensor Development (Beta)

Objectives

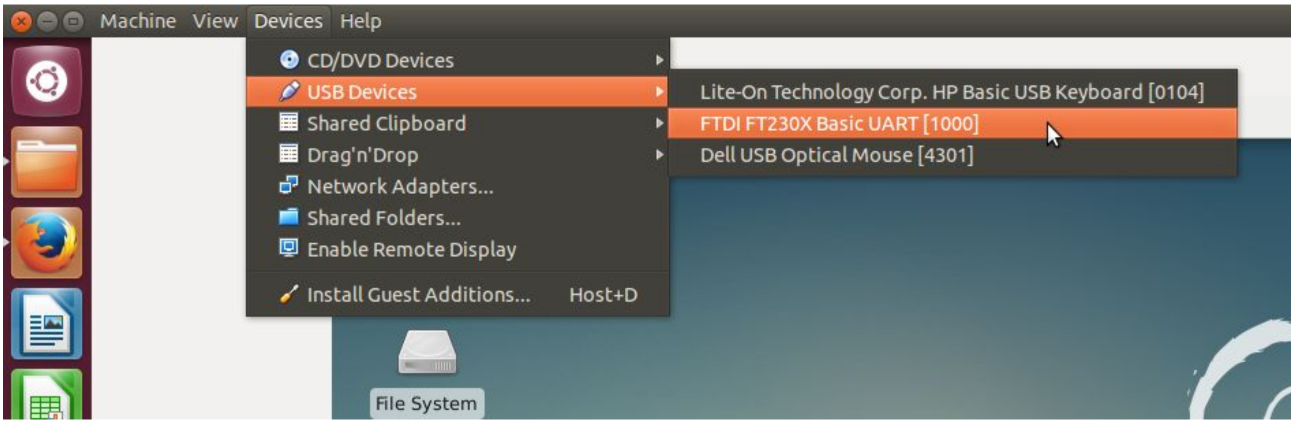
LoPy architecture and development environment (Atom IDE)
Sensors and Actuators (LED)
Data acquisition (1-wire)
Building a weather station (temperature)

Configuration

VirtualBox should be running after login, but if it is not, find and execute it from the main menu. Then execute virtual image of debian operating system specially prepared for this course. To run, just click on Start button. When you the reach login screen input credentials:

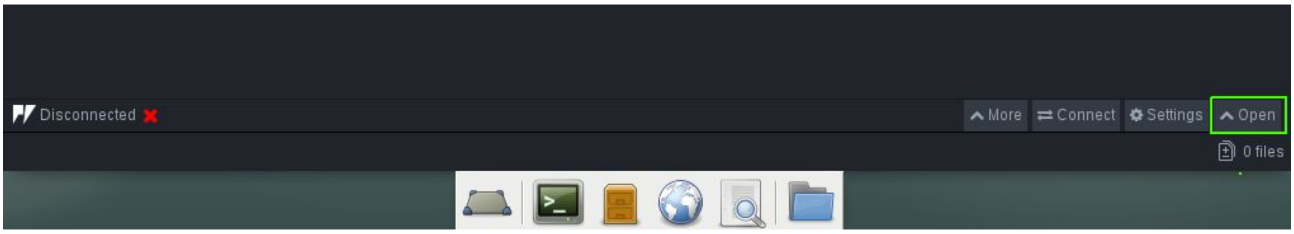
username: best
password: best

After logging into the VirtualBox image connect **Pycom** device, it is required giving permissions to VirtualBox to control USB devices connected to computer (Devices -> USB Devices -> FTDI FT230X Basic Uart)
Note: If you remove the USB Cable from computer you need to perform this step again.

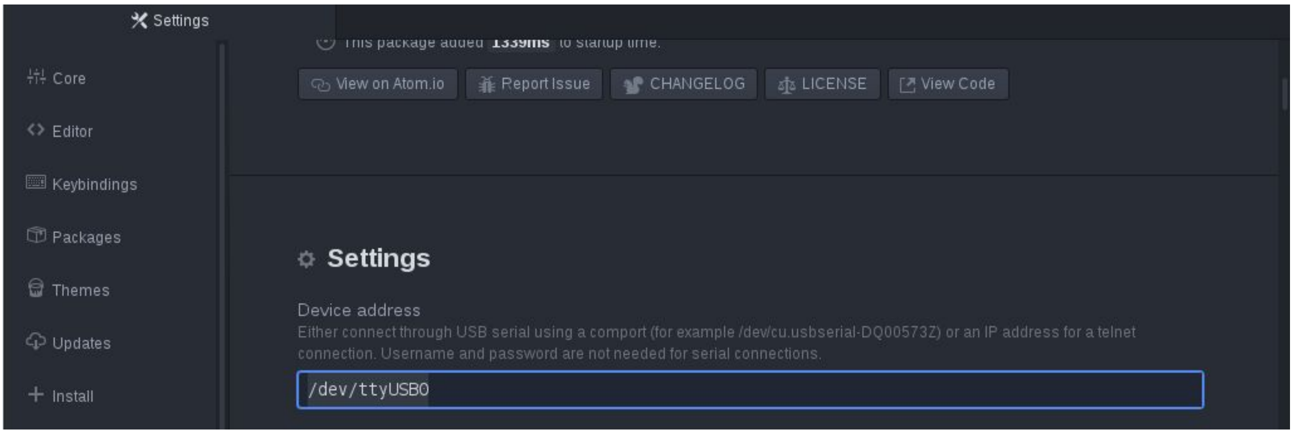

Machine
View
Devices
Help
CD/DVD Devices
USB Devices
Shared Clipboard
Drag'n'Drop
Network Adapters...
Shared Folders...
Enable Remote Display
Install Guest Additions... Host+D
Lite-On Technology Corp. HP Basic USB Keyboard [0104]
FTDI FT230X Basic UART [1000]
Dell USB Optical Mouse [4301]
File System

In order to communicate with a Pycom device via USB Serial, a few settings need to be configured. Below are the steps required to find, set and connect to a Pycom device over USB Serial.

- Ensure that the Pycom device is turned on and connected.
- Open Atom Editor located in the desktop.
- If the console is closed, click Open on the Pymakr Console located in the bottom, followed by the More tab.



- Next, click *Get serial ports*. This will copy the first serial port to the editor's clipboard.
- On the Pymakr Console Navigate to *Settings* -> *Global Settings*, scroll down and paste this address into the *Device Address* text field (should be this or similar: /dev/ttyUSB0)



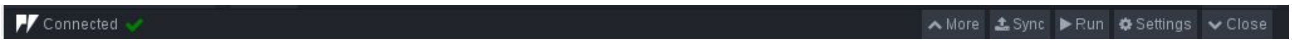
Settings

Device address

Either connect through USB serial using a comport (for example /dev/cu.usbserial-DQ00573Z) or an IP address for a telnet connection. Username and password are not needed for serial connections.

/dev/ttyUSB0

- Click Connect



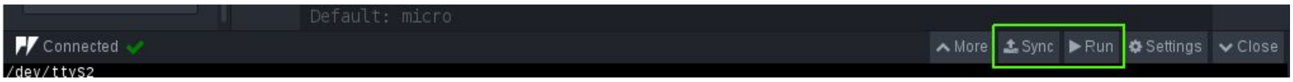
Tasks

1. Create a project that will contain all your files. Go to the top taskbar, File -> Add Project Folder and select the folder named “IoT-Sensors” located in your Desktop. The folder contains all necessary libraries for this laboratory.

2. Create a new file (File -> New file) and copy the code below. Try to understand the code because it will be used in future exercises. Save it (Ctrl + S) as **ex2.py** in the current folder. (lopy-lab1 located in your Desktop)

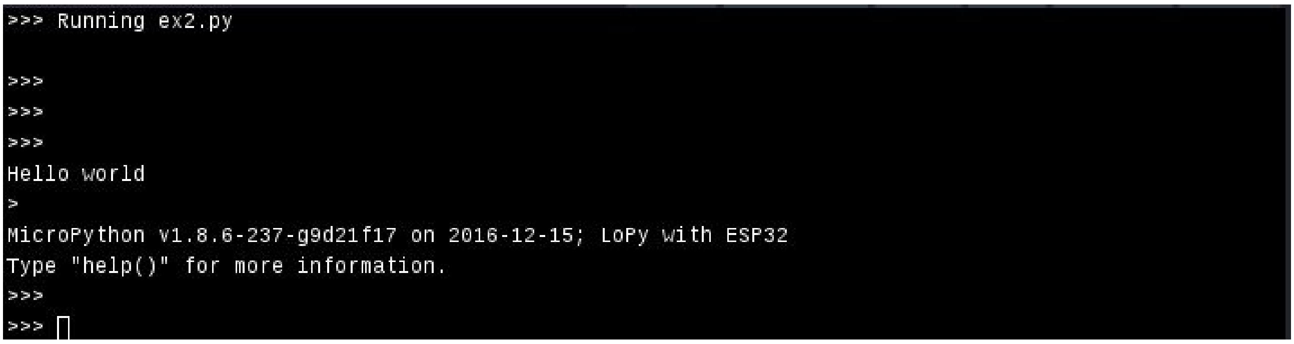
```
#import necessary libraries
import pycom
import time

#deactivates blinking LED present on the LoPy board
pycom.heartbeat(False)

#print Hello World in serial monitor
print("Hello world")
```

After writing and saving the code, there are 2 options to execute the code. You can save your file naming it main.py and use button ***Sync***, which will upload all files in your project to the Pycom device. Or use button ***Run*** that will only execute current code, in which case the filename does not matter. **Do not forget to save file before Run or Sync.**



Note: When LoPy is booted it will try to execute main.py file, if not found in memory it will execute default behavior, which is blinking the LED.

Expected results (the LED on the LoPy device stops blinking):



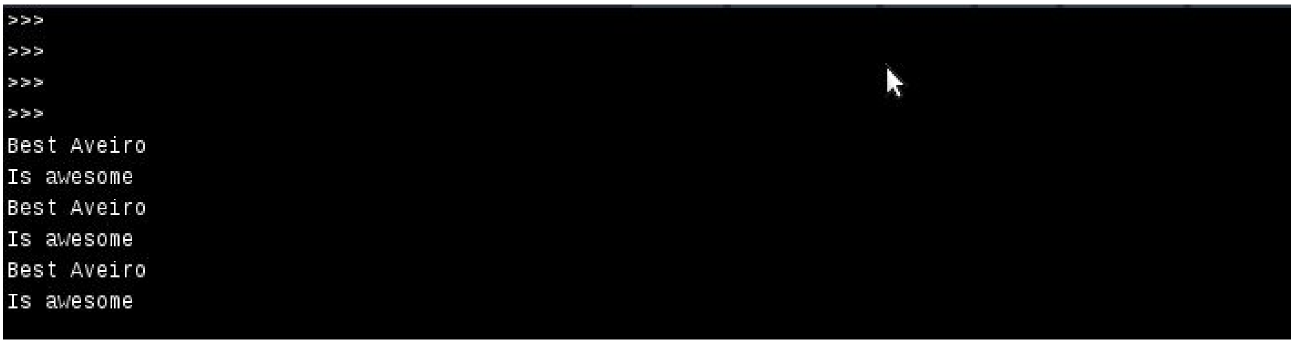
```
>>> Running ex2.py

>>>
>>>
>>>
Hello world
>
MicroPython v1.8.6-237-g9d21f17 on 2016-12-15; LoPy with ESP32
Type "help()" for more information.
>>>
>>> 
```

3. You can reuse previous exercise code, but save with a different name, **ex3.py** Try to define different LED colors using **pycom.rgbled()**, as argument you should use an RGB color code in hexadecimal or decimal format (example colors: Red (0xFF0000), Green (0x00FF00), Blue (0x0000FF)). It is also possible to combine colors for example White (0xFFFFFF), Orange (0xFF4500). *To completely deactivate LED use Black color code (0x000000)*

4. In a microprocessor such as the LoPy it is possible to delay events.

   Import library ***time*** and use commands such as **time.sleep**(1) to delay 1 second or **time**.**sleep_ms**(100) to delay 100 milliseconds, also use the ***while*** function with argument ***true***, which will repeat an infinite number of times.

   In this task write to the console “Best Aveiro” and 1 second after “Is awesome” in an infinite loop.



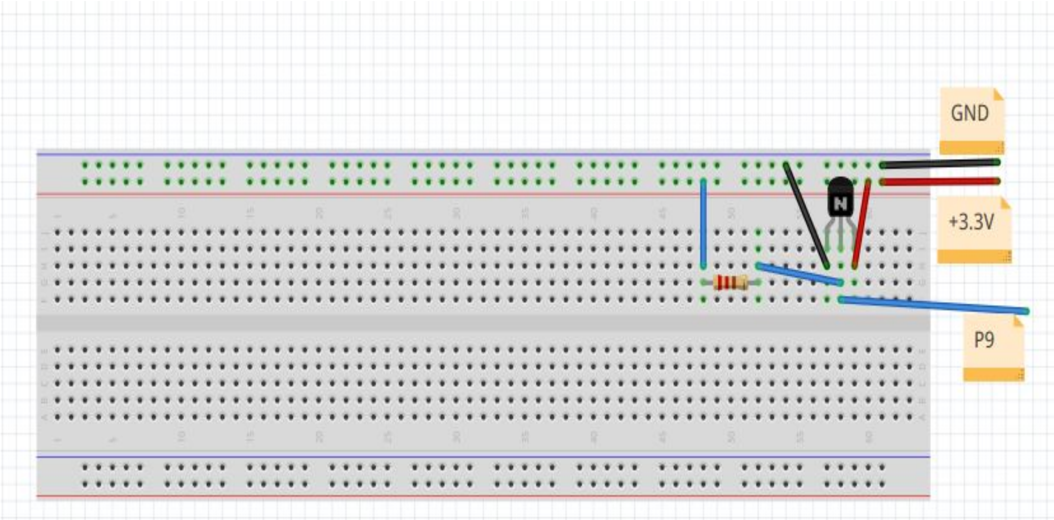
```
>>>
>>>
>>>
>>>
Best Aveiro
Is awesome
Best Aveiro
Is awesome
Best Aveiro
Is awesome
```

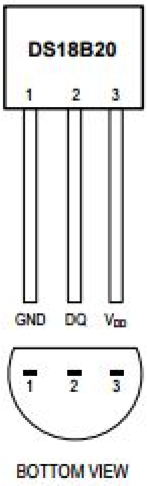
5. Using the knowledge obtained from the previous exercises, make the LoPy’s LED change from color BLUE to RED every 500 milliseconds.

6. The DS18B20 temperature sensor is a very popular 1-wire device, in this exercise we will teach you how to use the onewire module to read from such a device.

   As in all 1-wire bus, you must power the sensor and connect a 4.7k Ohm resistor between the data pin and the power pin.

Use the breadboard and supplied wires to assemble your circuit according to the following image:





After correctly assembling the circuit you need to program the LoPy.

You need to import the following libraries:

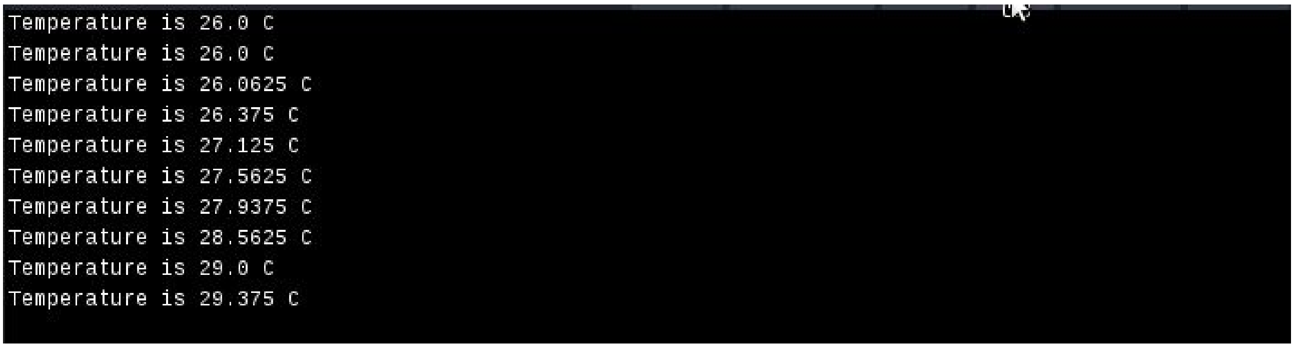
```
from machine import Pin
from onewire import DS18X20
from onewire import OneWire
```

Configure input pin as temperature reading pin:

```
ow = OneWire(Pin('P9'))
temp = DS18X20(ow)
```

You must execute the ***temp.start_convertion()*** function to initiate a temperature reading, then wait at least ***750ms*** before reading the value. In order to read use ***temp.read_temp_async()***

This exercise require importing OneWire library, use **Sync** button to upload library to Pycom device.



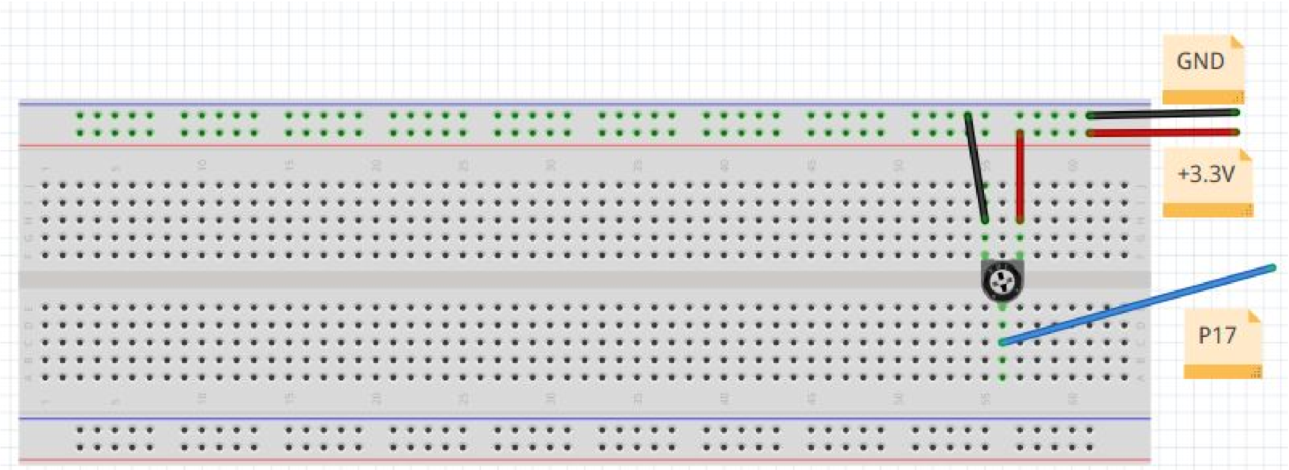
```
Temperature is 26.0 C
Temperature is 26.0 C
Temperature is 26.0625 C
Temperature is 26.375 C
Temperature is 27.125 C
Temperature is 27.5625 C
Temperature is 27.9375 C
Temperature is 28.5625 C
Temperature is 29.0 C
Temperature is 29.375 C
```

The task is to read and print the temperature value.

Extra task: Reading the value of a potentiometer using the ADC

Connect a potentiometer to P9 pin (we will use this pin to read voltage at the potentiometer). The reading can go from 0 to 1023 (10bit resolution).

Your task is to vary the LED intensity in the blue color. Low Blue (0x000001) - High Blue(0x0000FF) or from 1 to 255 in decimal.



Library to import:

from machine import ADC

Configure potentiometer as ADC device:

adc = ADC(0)
potentiometer = adc.channel(pin='P9')

Read potentiometer value:

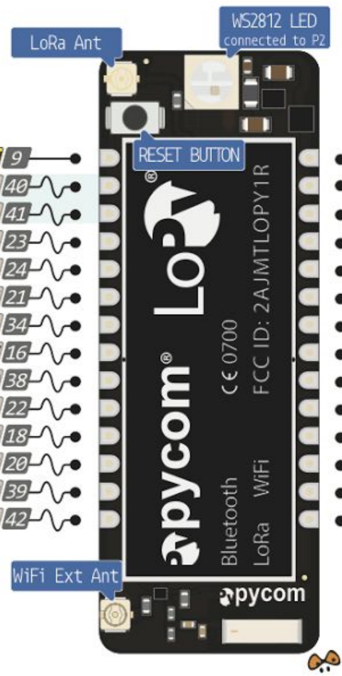
potentiometer.value()

LoPy® PINOUT

Internal Functions

35	GPIO18		VSPICLK	HS1DATA7	Lora Reset
25	GPIO16	EMACCLKOUT	U2RXD	HS1DATA4	External Antenna Switch
36	GPIO23		VSPIID	HS1STROBE	Lora Interrupt
27	GPIO17	EMACCO180	U2TXD	HS1DATA5	Lora Select

⚠ Absolute MAX per pin 12mA
recommended 6mA



PROGRAM Port: P0, P1

Connected to LoRa module ⚠: CLK, MOSI, MISO

RST									ChipPU	9
P0				CLKOUT2			U0RXD		GPIO3	40
P1				CLKOUT3			U0TXD	EMACRXD2	GPIO1	41
P2				CLKOUT1	Touch1	RTCIO11	ADC2_1	EMACTXCLK	GPIO0	23
P3	HS2DATA1	SDDATA1	HSPIHD		Touch0	RTCIO10	ADC2_0	EMACTXER	GPIO4	24
P4	HS2CMD	SDCMD	HSPICS0	MTD0	Touch3	RTCIO13	ADC2_3	EMACRXD3	GPIO15	21
CLK	HS1DATA6			VSPICS0				EMACRXCLK	GPIO5	34
MOSI					Touch7	RTCIO17	ADC2_7	EMACRXDV	GPIO27	16
MISO				VSPIQ			U0CTS	EMACTXD0	GPIO19	38
P8			HSPIWP		Touch2	RTCIO12	ADC2_2		GPIO2	22
P9	HS2DATA2	SDDATA2	HSPIQ	MTDI	Touch5	RTCIO15	ADC2_5	EMACTXD3	GPIO12	18
P10	HS2DATA3	SDDATA3	HSPIID	MTCK	Touch4	RTCIO14	ADC2_4	EMACRXER	GPIO13	20
P11				VSPIWP			U0RTS	EMACTXD1	GPIO22	39
P12				VSPIHD				EMACTXEN	GPIO21	42

5V										
GND										
3.3V										
17	GPIO14	EMACTXD2	ADC2_6	RTCIO16	Touch6	MTMS	HSPICLK	SDCLK	HS2CLK	P23
14	GPIO25	EMACRXD0	ADC2_8	RTCIO6		DAC_1				P22
15	GPIO26	EMACRXD1	ADC2_9	RTCIO7		DAC_2				P21
13	GPIO33		ADC1_5	RTCIO8	Touch8	XTAL32				P20
12	GPIO32		ADC1_4	RTCIO9	Touch9	XTAL32				P19
10	GPIO34		ADC1_6	RTCIO4		VDET1				P18
11	GPIO35		ADC1_7	RTCIO5		VDET2				P17
8	GPIO39	ADCPA	ADC1_3	RTCIO3		SensVN				P16
7	GPIO38		ADC1_2	RTCIO2		SensCN				P15
6	GPIO37		ADC1_1	RTCIO1		SensCP				P14
5	GPIO36	ADCPA	ADC1_0	RTCIO0		SensVP				P13

⚠ Up to 1.2-A Maximum Load Capability

⚠ P13–P18: Only Input pins!
No pullup internal resistence

- 🟥 Power
- ⬛ GND
- Serial Pin
- 🟩 Analog Pin
- 🟨 Control
- Physical Pin
- Port Pin
- Touch Pin
- 🟧 DAC Pin
- ∿ PWM Pin

Hack your LoPy!

Connect to a 10nF capacitor
to enable Touch Pin function

05 Nov 2016
ver 1 rev 1