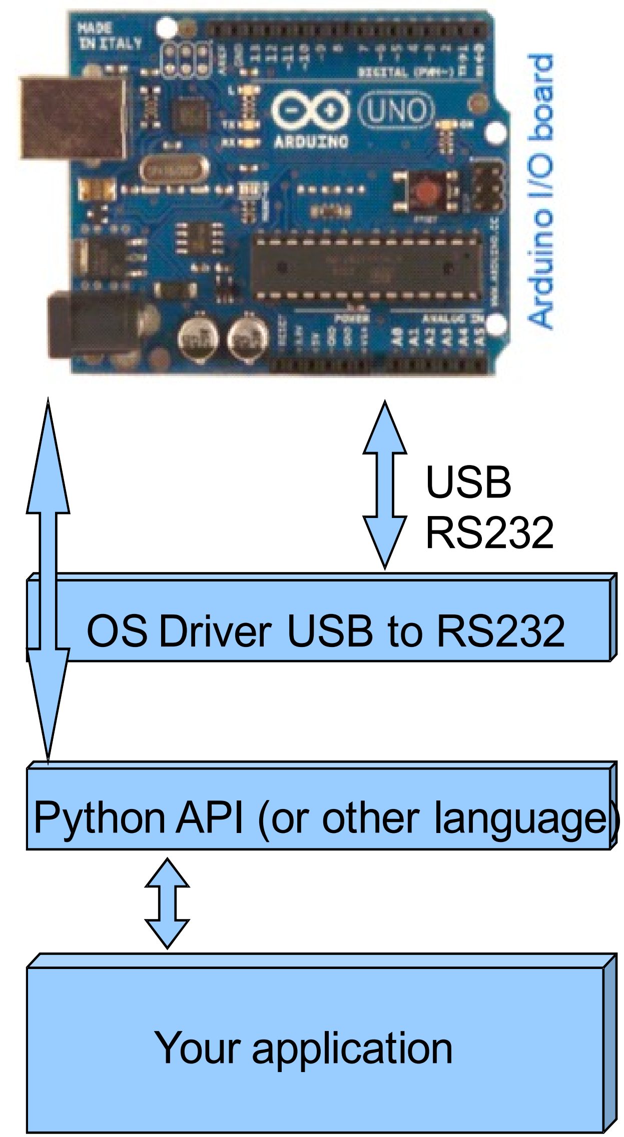
Lab. 10/B

Table of Contents

High-level communication protocol
design and implementation

- The **communication** between micro-controller based boards and computers is accomplished using **application level protocols** on top of low level data transmission interfaces (ex: RS232 or USB).
- Since a micro-controller is a ***general purpose controlling device,*** any high-level protocol design and implementation is left to the developer.
- Some high-level protocols are application specific while others are **more flexible** and can accommodate a **wide range of applications**.
- The purpose of this task is to **implement a flexible bi-directional communication protocol** for sensor-based micro-controller boards.
- The protocol will have the following **features**:
  - Read sensor values from a micro-controller board
  - Read component status values (ex: LEDs)
  - Send actions to the board (ex: turn on led)
-



High-level communication protocol
design and implementation

Communicating systems use **well-defined formats** for exchanging messages. Our protocol uses a **client server approach** where the computer is the client and the microcontroller is the server.

- ***The following data format is used for data exchange:***

Control Header	Sensor type (1)	Sensor data (1)	Sensor type (2)	Sensor data (2)	Sensor type (3)	Sensor data (3)

- ***The frame structure has the following meaning:***
  - **Control Header:** 8 bit field.
    - If the most significant bit is set to 0 then the frame is a GET request/reply, otherwise it's a SET request/reply;
    - The following bits indicates the number of sensor type-data blocks; If the frame is a SET reply then all the bits are set to 0.
  - **Sensor type:** 8 bit field.
    - If the most significant bit is 0 then the data field is 8 bits long, otherwise is 16 bits long;
    - The following bits indicates sensor/component number (ex: led 4).
  - **Sensor data:** 8 or 16 bit field.
    - sensor/component value for GET replies and SET requests;
    - Missing for GET requests and SET replies.

High-level communication protocol

instruction sheet

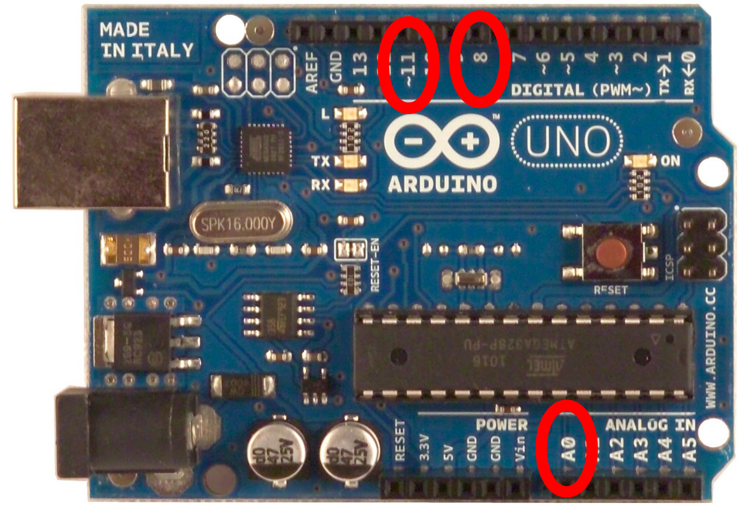
To Do:

- Change directory to "***Lab10b/Step_1***";
- Open the file "***hlcp.pde***" with Arduino IDE;
- Follow the instructions above the "TO DO" placeholders and fill in missing lines. Then compile and upload the program onto the micro-controller board;
- Open the file "***hlcp.py***" with Notepad++;
- Follow the instructions above the "TO DO" placeholders and fill in missing lines. Then compile and run the program using the python interpreter;
- Observe the output and explain the whole process of fetching data from a sensor, data transfer to the computer and data displaying on the computer screen;
- Try to extend the programs to include information from the light sensor.

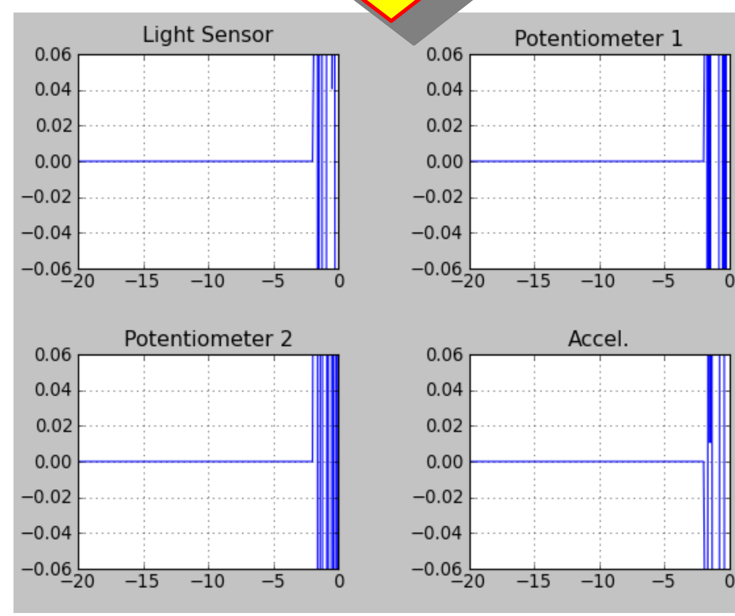
Lab. 10/B
Table of Contents

Real-time monitoring display
using HLCP

- To make use of the **HLCP** communication protocol we've implemented, we will adjust a user interface to display sensor values in real-time;
- The display is developed in **python** and makes use of the **matplotlib library** to plot real-time graphs;
- Every second a GET request is sent by the monitoring application requesting data about the accelerometer (X, Y and Z axis) and potentiometer values;
- For every GET request a GET reply is sent by the micro-controller board with values for all requested sensors.
- The information is displayed using a buffer for the last 20 seconds.
- Additional buttons can trigger various actions on the micro-controller board side (such as turning a led on)



Arduino I/O board



Real-time monitoring display

instruction sheet

To Do:

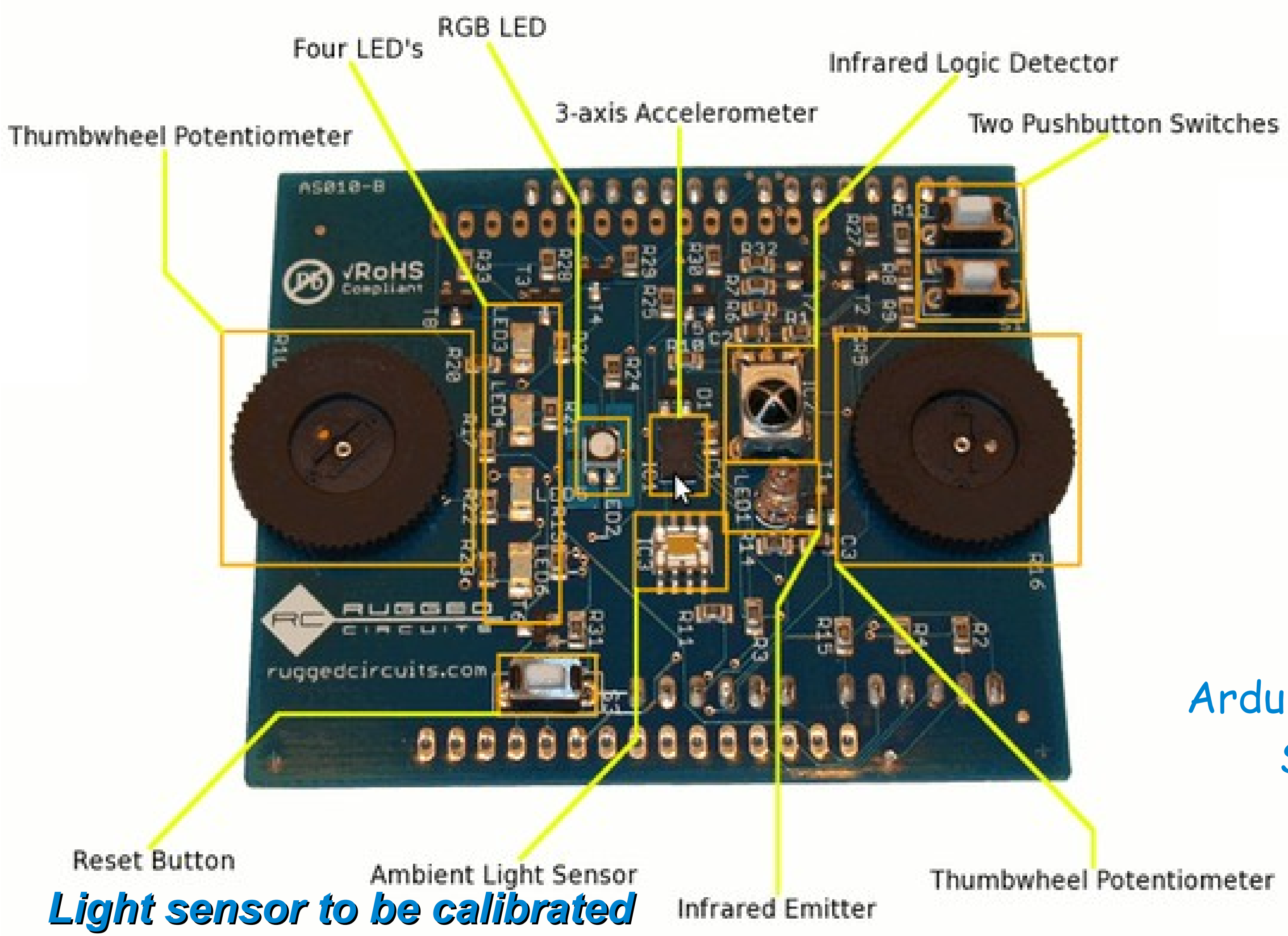
- Change directory to "***Lab10b/Step_2***";
- Open the file "***rtdisplay.py***" with Notepad++;
- Adjust the program to make us of the previously implemented high level protocol;
- Follow the instructions above the "TO DO" placeholders and fill in missing lines;
- Extend the program to send a SET request to the micro-controller board whenever the "Flip led state" button is pressed. The request should turn LED 1 off if it's on and turn it on if it's off;
- Extend the program to act as a trigger and turn on LED 1 whenever the potentiometer 1 value is greater than 80 out of 100.

Lab. 10/B
Table of Contents

Sensor calibration
light sensor

Plan of action:

Calibrate the light sensor using the artificial room light as a maximum and some custom chosen intermediary references.



Light sensor to be calibrated

Arduino Gadget Shield

Sensor calibration

light sensor

To Do:

- Change directory to “***Lab10b/Step_3***”;
- Compile and upload “***hlcp.pde”*** program onto the micro-controller board if the board is not already programmed;
- Make use of the real-time monitoring display to calibrate the light sensor. Consider the artificial ambient light the maximum light intensity, then choose some intermediate references and compute the calibration formula.
- Open the file “***hlcp.pde***” with Arduino IDE;
- Adjust the program to use the new light sensor calibrated value instead of the default one;
- Check the result using the real-time monitoring display;
- Try to implement the same calibration but in software (python program) rather than on the controller side.

Lab. 10/B
Table of Contents

Optional homework

- Use the “Ethernet shield” to implement the same **hlcp** protocol to communicate using **TCP/IP instead of RS232**;
- Adapt the **real-time monitoring display** to connect to a certain IP and receive all the data using **hlcp** via TCP.