Snap! and Arduino

01. Simple LED Control

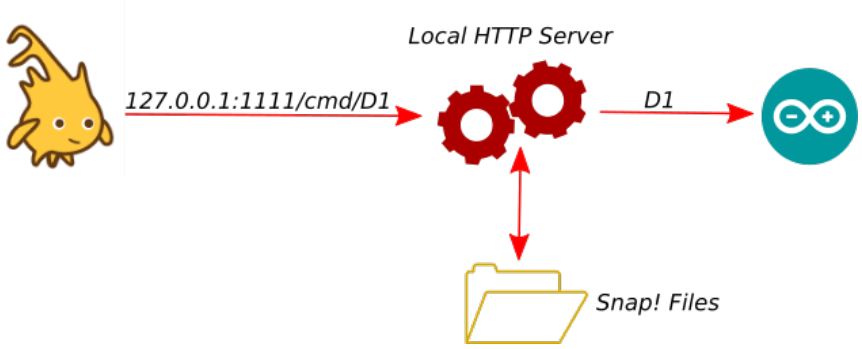
https://github.com/pfabo/snap_and_world

1. decembra 2019

Kapitola 1

01. Snap! and Arduino

We will use Arduino as a simple input/output device for data collection and control of simple experiments in Snap!. For security reasons, Snap! communicate directly only with standard computer peripherals (keyboard, mouse, sound system, camera ...). For communication with the outside world we will use a simple HTTP server writen in Python, which will form a bridge between our program and peripherals.



Obr. 1.1: Snap! and HTTP Server

In first simple example, we will use the Snap! to control the Arduino on-board LED. All source codes are available on github https://github.com/pfabo/snap_and_world

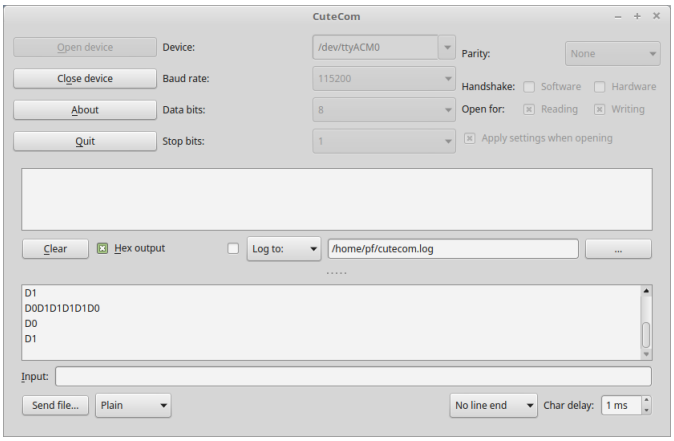
1.1 Arduino Client Program

For Arduino we will create a simple program that will receive characters from the serial interface and control the on-borad LED.
If the program receives a sequence of 'D1' characters, the LED lights up, and the 'D0' sequence turns LED off.

```
#include <Arduino.h>

char buffer[2];
int byteReads=0;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);                    // init on-board LED
  Serial.begin(115200);                            // init serial comm
}

void loop() {
  if (Serial.available()){
    byteReads = Serial.readBytes(buffer, 2);       // check if data has been sent
    from the computer

    if (byteReads == 2 && buffer[0] == 'D'){       // parse data
      if((buffer[1] - '0') == 1){                  // command 'D1'
          digitalWrite(LED_BUILTIN, HIGH);
      }
      else{                                        // command 'D0'
          digitalWrite(LED_BUILTIN, LOW);
      }
    }
  }
}
```

1.1.1 Check

Connect arduino to your computer, upload program to board via programming environment and run some terminal emulator (check port name and baudrate). Send sequence D1 and D0 to Arduino, on-board LED should be on and off.



Obr. 1.2: CuteCom Terminal

1.2 Simple HTTP Server

Edit the port name in the server source code to the port name where your Arduino is connected.

```
import serial
import struct
from socketserver import ThreadingMixIn
from http.server import HTTPServer, SimpleHTTPRequestHandler

port = None
port_name = '/dev/ttyACM0'   # /dev/ttyUSB0 ...

class BoardHandler(SimpleHTTPRequestHandler):

    def log_message(self, format, *args):
        return

    def do_GET(self):
        global port
        s = self.path.split('/')
        if 'cmd' in s:
            for c in s[2]:
                port.write(struct.pack('B', ord(c)))

            resp = 'ok'
            message = resp.encode()
            self.request.send(message)
        else:
            SimpleHTTPRequestHandler.do_GET(self)


class ThreadedHTTPServer(ThreadingMixIn, HTTPServer):
    pass

def http_server():
    try:
        server = ThreadedHTTPServer(('127.0.0.1', 1111), BoardHandler)
        server.daemon_threads = True
        print( "HTTP Server at port 1111, ^C to exit")
        server.serve_forever()
    except KeyboardInterrupt:
        server.socket.close()


if __name__ == '__main__':
    port = serial.Serial(port_name, 115200)
    http_server()
```

Start server

```
python server.py
```

1.2.1 Check

In your www browser enter address

```
127.0.0.1:1111/cmd/D0
```

and

```
127.0.0.1:1111/cmd/D0
```

Arduino on-board LED should switch between on and off state.

1.3 Snap!

Download Snap! program as mentioned on https://snap.berkeley.edu/offline
Unpack program in your directory with server code, the directory structure is then

```
MyDirectory
   |
   +--- server.py        <- HTTP server source code
   +--- index.http       <- redirector file (from github)
   |
   +--- snap.html        <- from Snap! install
   +--- Snap-5.1.0       <- Snap! code
          +---
          +---
          ...
```

1.3.1 Check

Connect Arduino board to your computer, check port name in *server.py* and start HTTP server. In your www browser enter address

```
127.0.0.1:1111
```

and local copy of Snap! should be loaded.

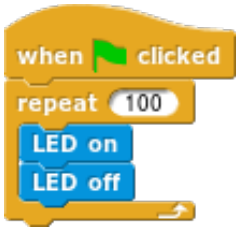
1.4 Snap! Programming

Create new blocks

Obr. 1.3: Block LED 0n

Obr. 1.4: Block LED 0ff

and enjoy new level of programming :-)



Obr. 1.5: Simple program