СОЗДАНИЕ СЕТЕВОЙ ИНФРАСТРУКТУРЫ ДЛЯ ПОДДЕРЖКИ ИННОВАЦИОННОГО
ПРЕДПРИНИМАТЕЛЬСТВА МОЛОДЕЖИ НА ПЛАТФОРМАХ ПРОИЗВОДСТВЕННЫХ ЛАБОРАТОРИЙ

FABLAB

DEVELOPMENT OF A NETWORK INFRASTRUCTURE
FOR YOUTH INNOVATION ENTREPRENEURSHIP SUPPORT ON FABLAB PLATFORMS

Co-funded by the
Erasmus+ Programme
of the European Union

Rapid development of a connected device

http://fablab-erasmus.eu

VNIVERSITAT DE VALÈNCIA

Escola Tècnica
Superior
d'Enginyeria

Electronics and Fablabs

ECOISME

HIDEEZ

6 of 24 most successful Ukrainian startups are making electronic devices

UniExo

Universal Medical Robotized Exoskeleton

Petcube

iblazr

Electronics and Fablabs

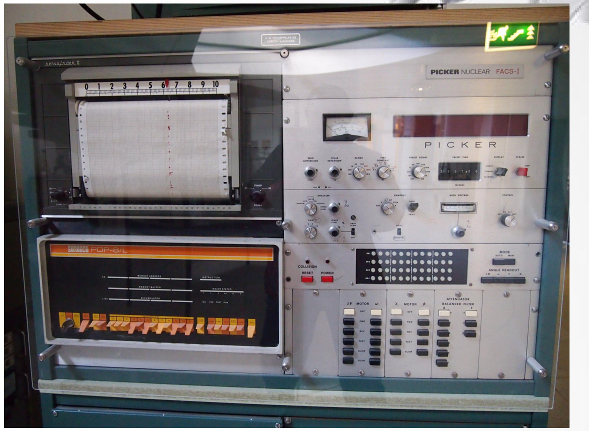
- Open Source Hardware market costs more than 1 billion $
- 2015: UK computer literacy project; Every 11-12 year old child can get a Micro Bit electronics kit

Embedded Electronics

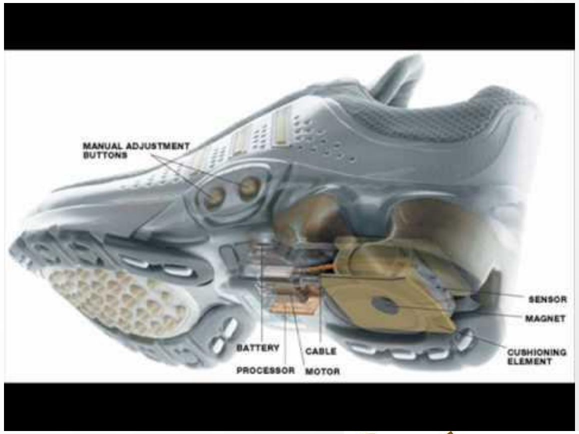
- 1970s – Integrated Circuits become affordable for small business and amateurs
- 1990s – Microcontroller revolution. Small automation goes digital
- 2000s – Arrival of cheap and simple MCU development tools on the market.
- 2010s – Systems-on-Chip with radio become widespread. Drastic reduction of time and costs for the connected device development

Embedded electronics

- 1976: The whole computer have to be *embedded* into the device

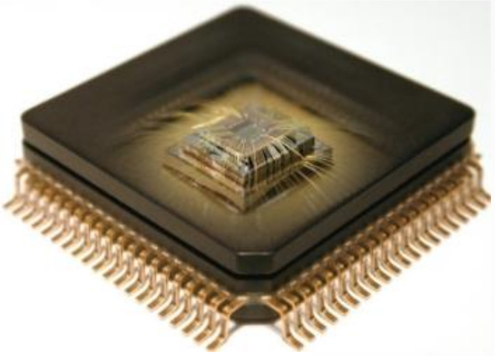


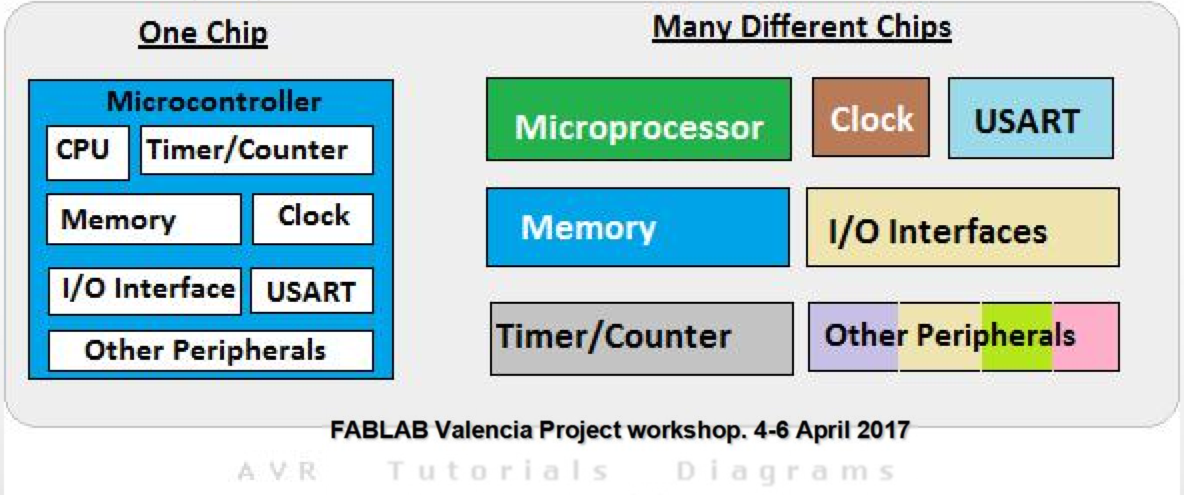
- 2016: the computer fits into a single chip and is embedded into a shoe



Microcontroller

- A self-suffucient integrated circuit that can be programmed and execute the program with the minimum amount of external components.
- Due to universality, microcontrollers are mass-produced and usually are cheaper than discrete component systems
- 1990s: the silicon becomes cheaper than the packaging





One Chip

Microcontroller

CPU

Timer/Counter

Memory

Clock

I/O Interface

USART

Other Peripherals

Many Different Chips

Microprocessor

Clock

USART

Memory

I/O Interfaces

Timer/Counter

Other Peripherals

Development process 1

- Existing problem or new functionality added
- Device concept
- Technologies for the desired functionality
- Contradicting factors and compromises

Development process 2

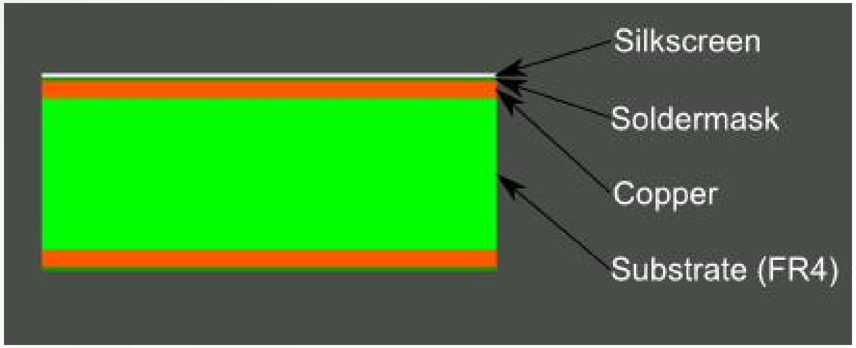
- Parts and tools necessary
- Prototyping
- Debugging of the prototype. Idea evaluation.
- Preparing the prototype for the mass production

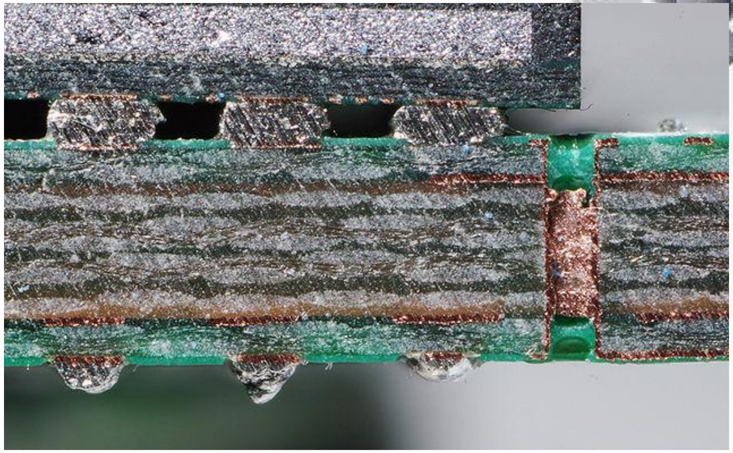
Prototype creation

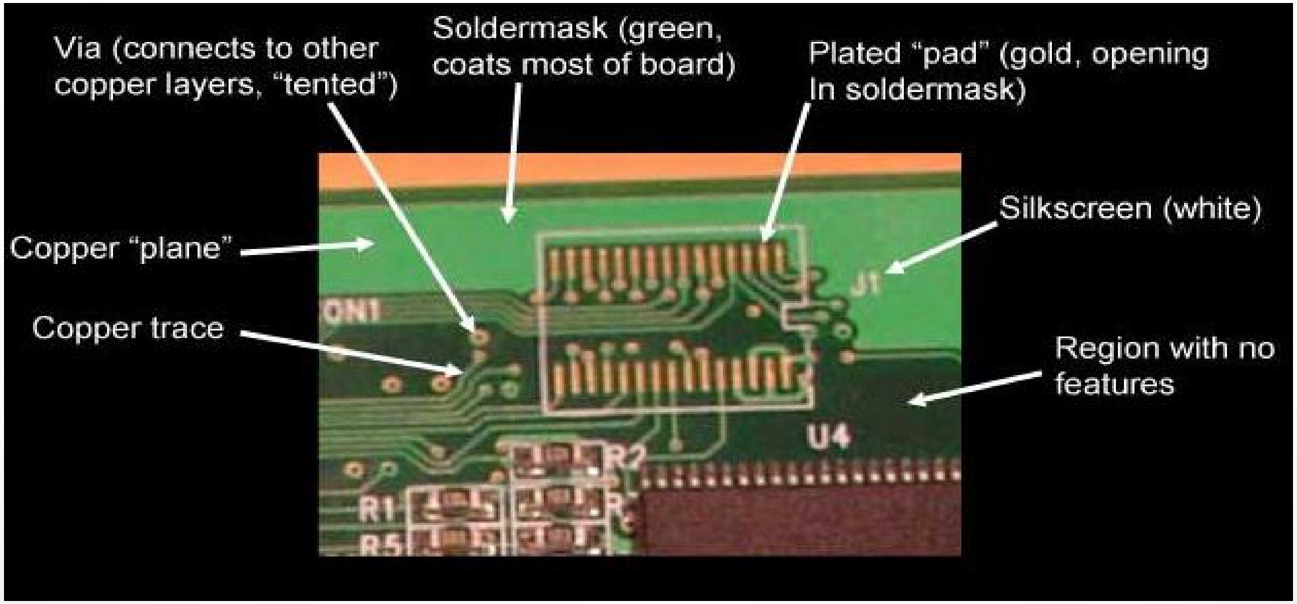
- Block schematic of the device
- Component selection
- CAD Schematic design
- PCB design
- PCB manufacturing
- PCB soldering
- MCU programming
- Testing

VNIVERSITAT DE VALÈNCIA

PCB anatomy


Silkscreen
Soldermask
Copper
Substrate (FR4)




Via (connects to other copper layers, "tented")
Soldermask (green, coats most of board)
Plated "pad" (gold, opening In soldermask)
Silkscreen (white)
Copper "plane"
Copper trace
Region with no features
ON1
J1
U4
R1
R2
R5

PCB design

- “Device design first” - the mechanical dimensions and the shape of the PCB must be defined beforehand.
- Connectors, board mount elements: placement, size, shape...
- Heat
- EMC/crosstalk

PCB manufacturing - hobby

- Engraving
  - + Fast, cheap
  - + No chemicals used
  - − Noisy and dusty
  - − Low resolution (typically >0.6 mm)
- Toner Transfer (LUT)
  - + Cheap
  - + Average resolution (up to 0.2 mm)
  - − Requires skill, results may vary
  - − Lots of manual work

PCB manufacturing - industrial

- Photoresistive method

  + Best detalization (up to 0.05 mm)

  + Manufacturing of high complexity PCBs

  + High repetibility

  + Mass production ready

  - The machinery is very expensive (but there are solutions for home manufacturing too)

  http://pcbshopper.com/

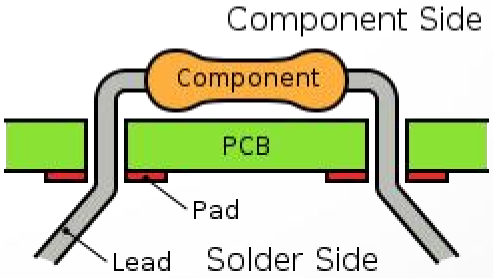
Component selection checklist

- Availability
- Documentation and manufacturer support
- Price
- Previous experience
- Mounting technology available

All the forementioned (and some more) parameters necessary to be taken into account when choosing the right ones!

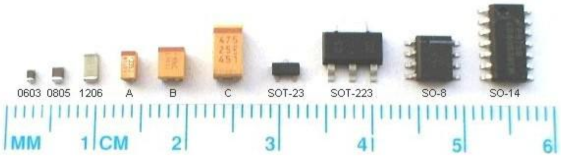
Components: packaging

- THT (through-hole technology)
  - Large
  - Easier to mount for inexperienced users
  - More expensive in mass production
  - Some elements available only in THT



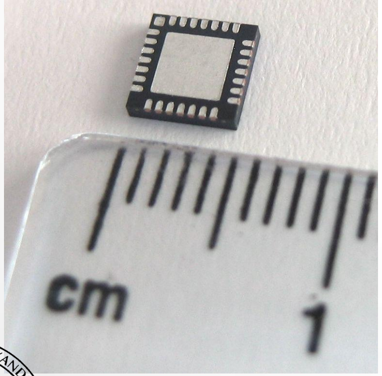
Components: packaging

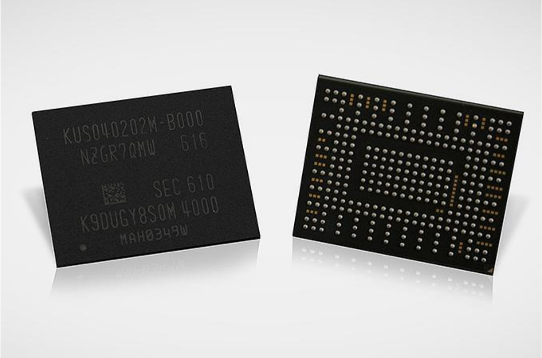
- SMD (surface mount)
  - Can be very small
  - Cheaper components, cheaper board manufacturing
  - Harder for manual assembly; special tools sometimes required



Components: packaging

- BGA or other no-leads package
  - Smallest components and highest density on the board
  - The only solution for high-end complex ICs, like processors
  - Requires special equipment to solder



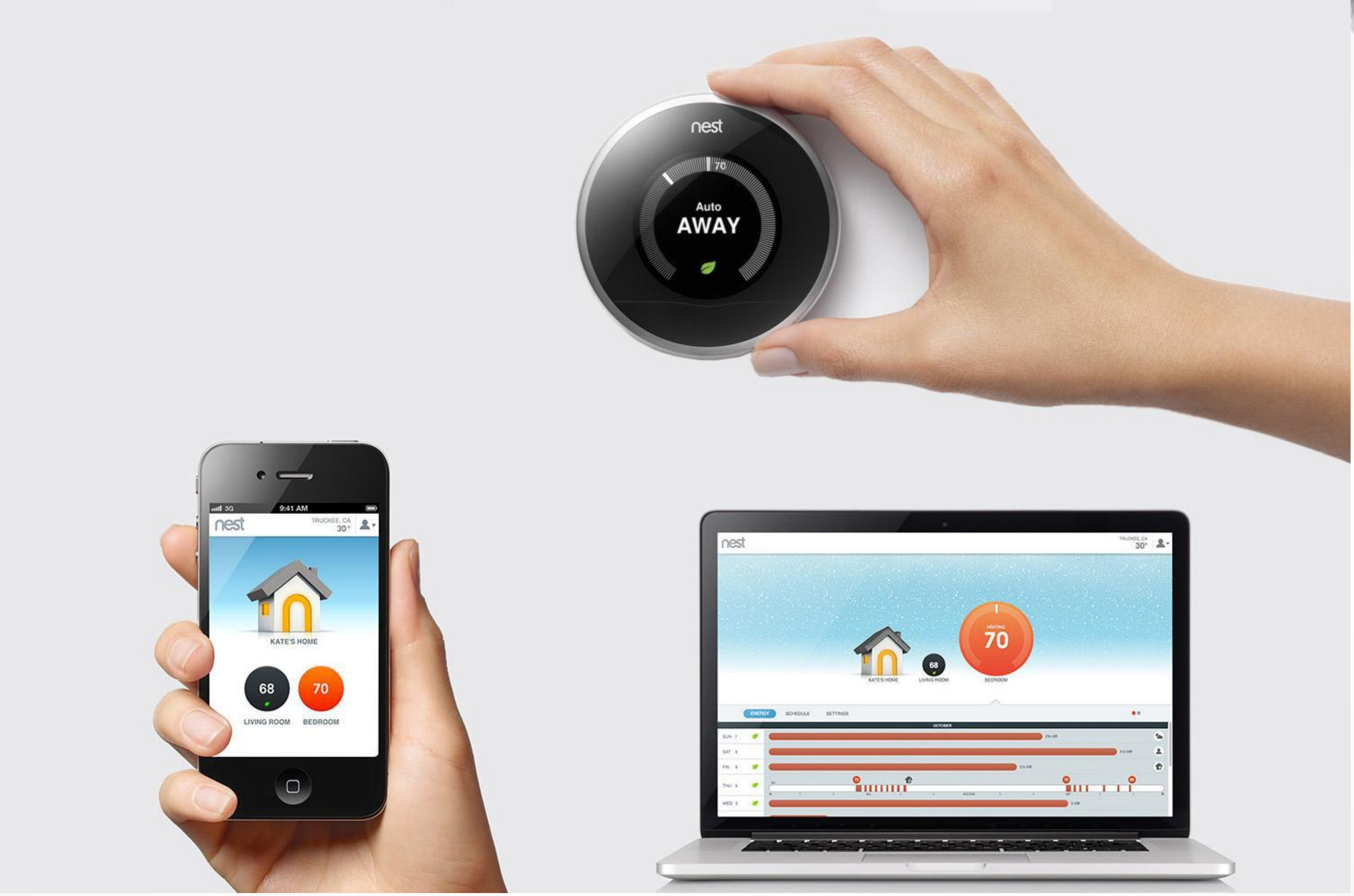


Components: sources

Let's try to find and compare parts for the digital thermometer IC on the next websites:

- mouser.es
- www.farnell.com
- http://www.digikey.com/

The device: Idea

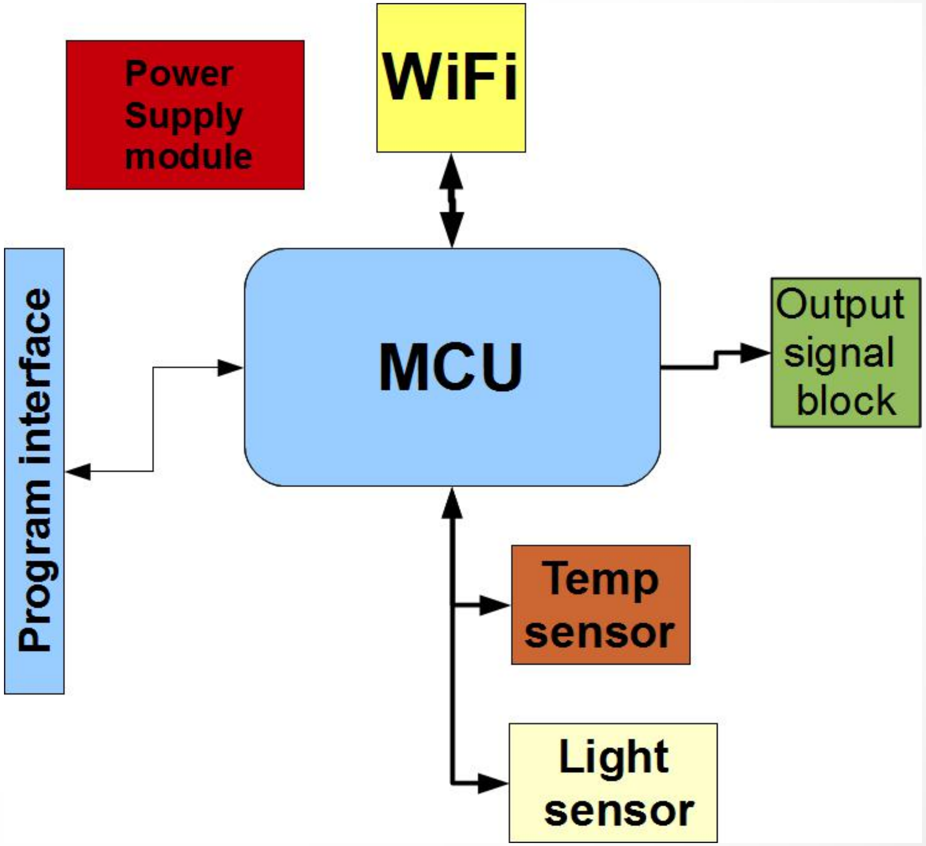


The device: guidelines

- Wi-Fi enabled programmable thermostat/light sensor
- Power: always on
- Power source: external
- Inputs: temperature sensor, luminosity sensor
- Outputs: single on/off output.
- Programmable via computer/smarthone

Block schematic

- Power
- Processor
- Wireless comm
- Sensors
- Output



Component selection

- Power: cheap linear regulator. Input: 5-9V, output 3.3V stable
- MCU+Radio: ESP8266 SoC. Integrated WiFi solution
- Light sensor: cheap photoresistor
- Temperature sensor: digital inexpensive LM75 thermometer

ESP8266

- An integrated solution, containing WiFi module and MCU in a single chip
- Cheap (<2.5$) and easy to obtain
- Large amount of development tools, big community
- Modules available, removing the necessity to design RF part (hard one)
- Firmware libraries available

VNIVERSITAT
DVALÈNCIA

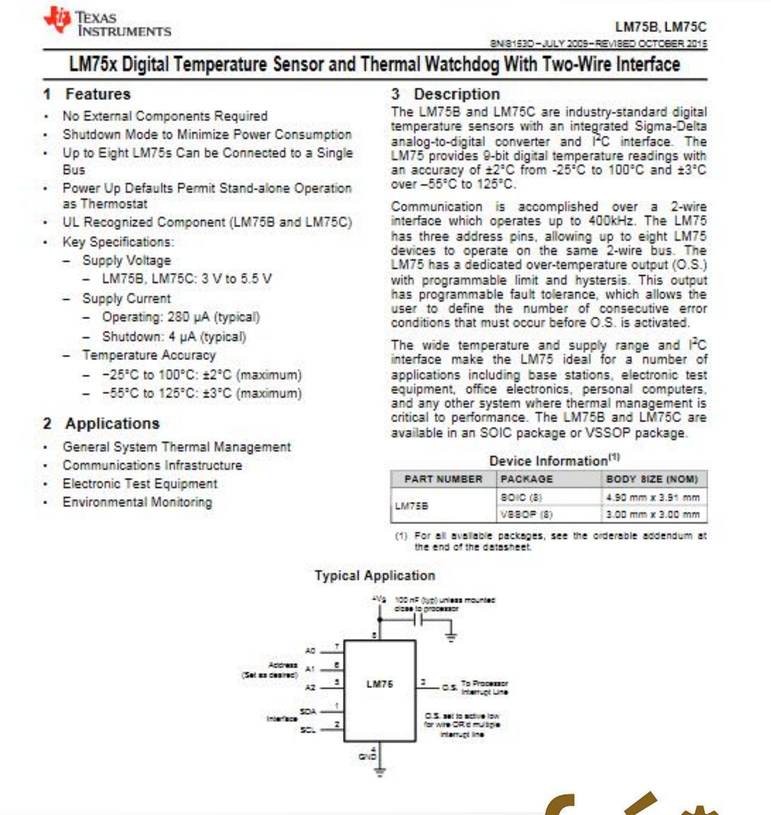
Engineer's best friend: Datasheet

Every component manufacturer provides an extensive set of documentation on its production.

Types of documentation usually provided:

- Datasheet
- Application note
- Instruction manual (for processors)
- Design guidelines

It is important to keep the latest copies of the parts' documentation with the project, many costly errors can be avoided by that



TEXAS INSTRUMENTS

LM75B, LM75C

SNIS153D – JULY 2009 – REVISED OCTOBER 2015

LM75x Digital Temperature Sensor and Thermal Watchdog With Two-Wire Interface

1 Features

- No External Components Required
- Shutdown Mode to Minimize Power Consumption
- Up to Eight LM75s Can be Connected to a Single Bus
- Power Up Defaults Permit Stand-alone Operation as Thermostat
- UL Recognized Component (LM75B and LM75C)
- Key Specifications:
  - Supply Voltage
    - LM75B, LM75C: 3 V to 5.5 V
  - Supply Current
    - Operating: 280 µA (typical)
    - Shutdown: 4 µA (typical)
  - Temperature Accuracy
    - −25°C to 100°C: ±2°C (maximum)
    - −55°C to 125°C: ±3°C (maximum)

2 Applications

- General System Thermal Management
- Communications Infrastructure
- Electronic Test Equipment
- Environmental Monitoring

3 Description

The LM75B and LM75C are industry-standard digital temperature sensors with an integrated Sigma-Delta analog-to-digital converter and I²C interface. The LM75 provides 9-bit digital temperature readings with an accuracy of ±2°C from -25°C to 100°C and ±3°C over –55°C to 125°C.

Communication is accomplished over a 2-wire interface which operates up to 400kHz. The LM75 has three address pins, allowing up to eight LM75 devices to operate on the same 2-wire bus. The LM75 has a dedicated over-temperature output (O.S.) with programmable limit and hystersis. This output has programmable fault tolerance, which allows the user to define the number of consecutive error conditions that must occur before O.S. is activated.

The wide temperature and supply range and I²C interface make the LM75 ideal for a number of applications including base stations, electronic test equipment, office electronics, personal computers, and any other system where thermal management is critical to performance. The LM75B and LM75C are available in an SOIC package or VSSOP package.

Device Information[1]

PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM75B	SOIC (8)	4.90 mm x 3.91 mm
	VSSOP (8)	3.00 mm x 3.00 mm

[1] For all available packages, see the orderable addendum at the end of the datasheet.

Typical Application

+Vs
100 nF (typ) unless mounted close to processor
8
A0
7
Address (Set as desired)
A1
6
A2
5
LM75
3
O.S.
To Processor Interrupt Line
O.S. set to active low for wire OR'd multiple interrupt line
SDA
1
Interface
SCL
2
4
GND

Schematic and PCB development

Electronics Design Automation - EDA

- Schematic creation
- Circuit simulation (not in all EDA)
- Printed circuit board creation
- Electrical and design rule check
- Generation of Gerber files (data format for industrial fabrication)
- Autogeneration of documentation.

EDA - programs and features

CAD Eagle (now part of Autodesk)

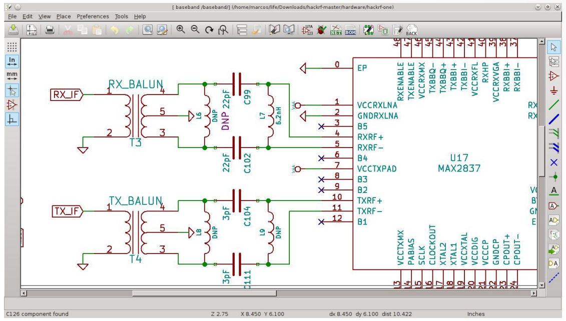
- Free version available, includes 2 schematic sheets, 2 signal layers, and 80 cm^2 board area.

KiCAD - free, open-source EDA

- Schematics of arbitrary complexity.

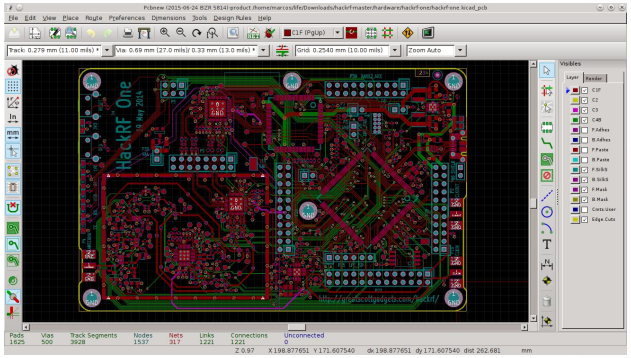
EDA Workflow

- Drawing the electrical schematic.


[baseband /baseband/] (/home/marcos/life/Downloads/hackrf-master/hardware/hackrf-one)

C126 component found   Z 2.75   X 8.450 Y 6.100   dx 8.450 dy 6.100 dist 10.422   Inches

- Printed Circuit Board design.

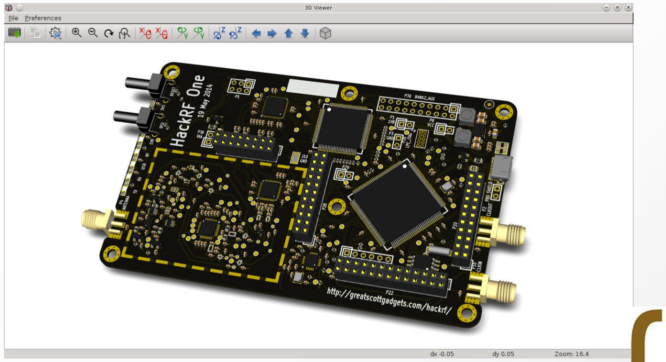

Pcbnew (2015-06-24 BZR 5814)-product /home/marcos/life/Downloads/hackrf-master/hardware/hackrf-one/hackrf-one.kicad_pcb

Track: 0.279 mm (11.00 mils) *   Via: 0.69 mm (27.0 mils)/ 0.33 mm (13.0 mils) *   Grid: 0.2540 mm (10.00 mils)   Zoom Auto

Pads	Vias	Track Segments	Nodes	Nets	Links	Connections	Unconnected
1625	500	3928	1537	317	1221	1221	0

Z 0.97   X 198.877651 Y 171.607540   dx 198.877651 dy 171.607540 dist 262.681   mm

- Optional: 3D view generation.

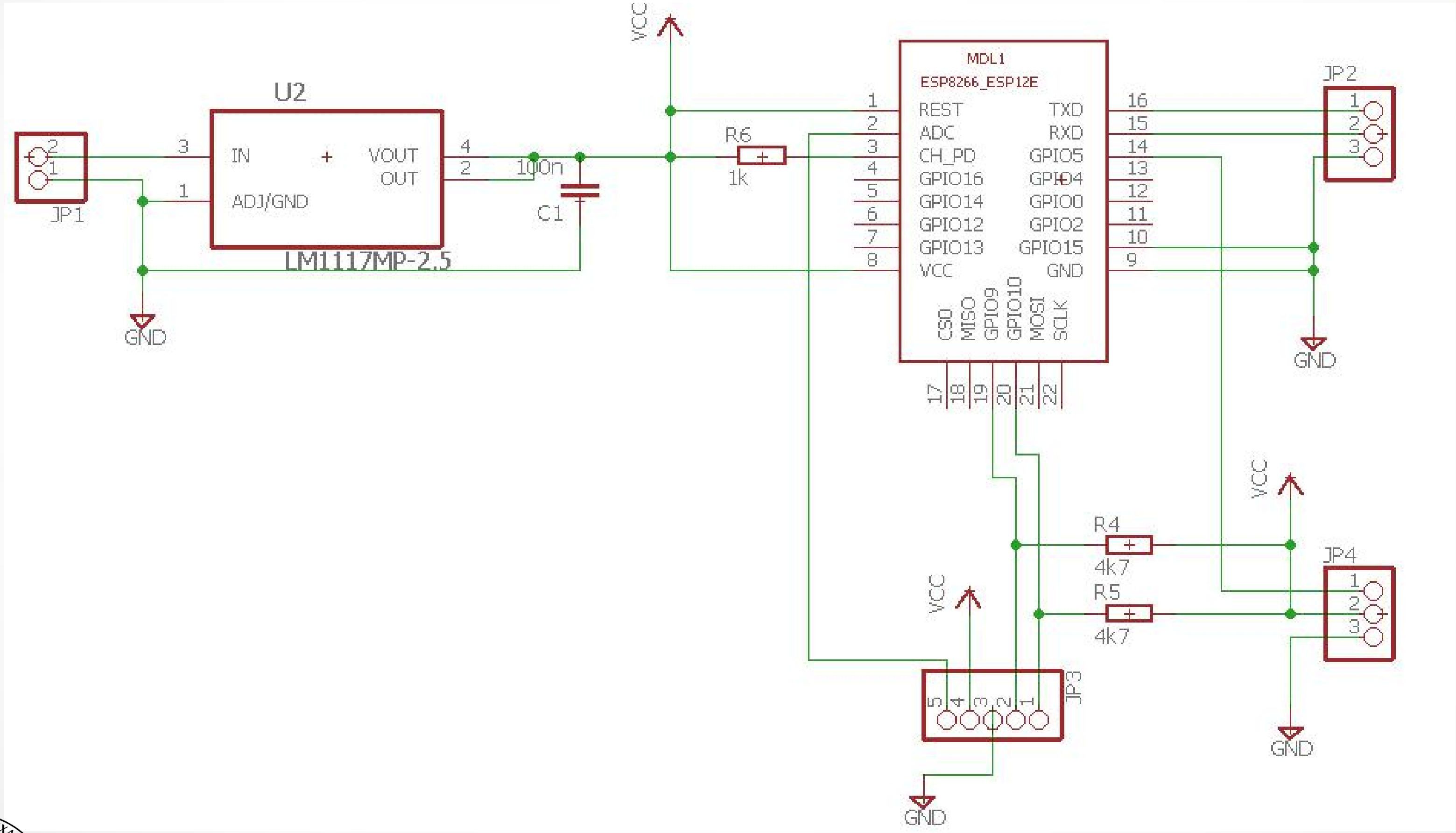

3D Viewer

dx -0.05   dy 0.05   Zoom: 16.4

Schematic drawing

- Prepare the libraries for the all components (and check them!)
- Place the elements on the schematic
- Draw connections between them
- ERC (Electric Rule Check)

Main board



U2
IN
VOUT
OUT
ADJ/GND
LM1117MP-2.5
JP1
100n
C1
GND
VCC
R6
1k
MDL1
ESP8266_ESP12E
REST
ADC
CH_PD
GPIO16
GPIO14
GPIO12
GPIO13
VCC
TXD
RXD
GPIO5
GPIO4
GPIO0
GPIO2
GPIO15
GND
CS0
MISO
GPIO9
GPIO10
MOSI
SCLK
JP2
JP3
JP4
R4
4k7
R5
4k7

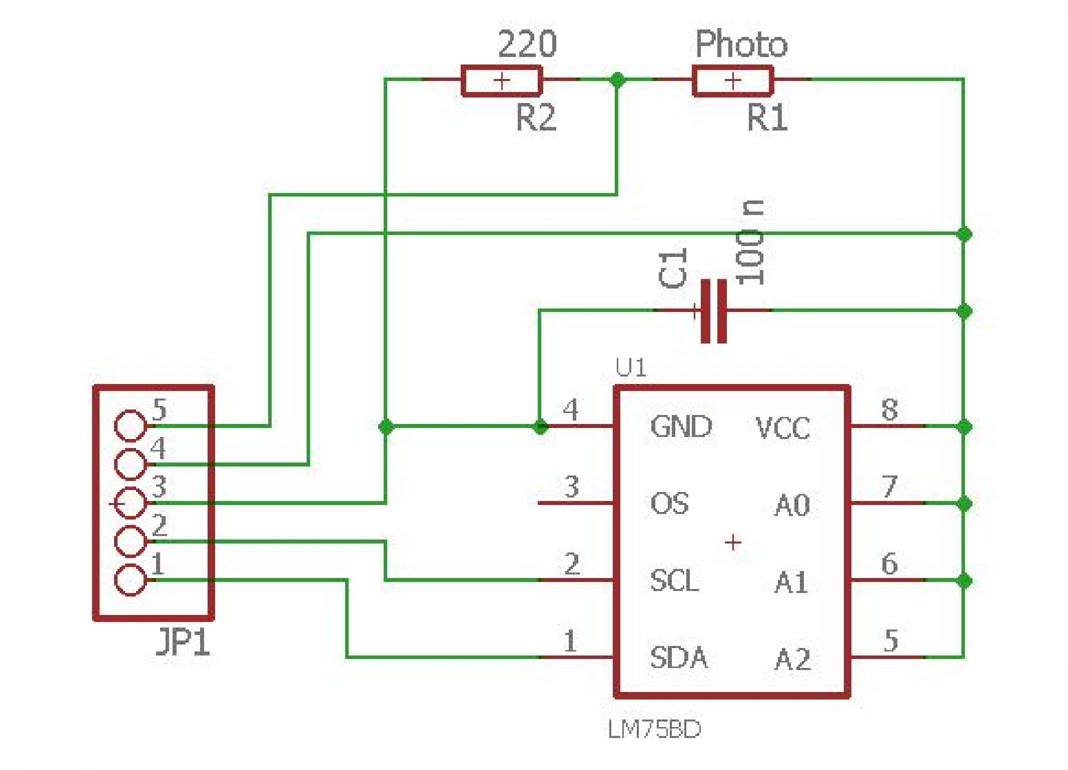
EDA concepts

- **Element -** a physically present component of the design. The element consists of **schematic symbol** and **footprint.** In complex EDAs it may also include the 3D model and electrical simulation model (SPICE).

- **Signal -** a concept of virtual wire. Sygnal is defined by its name and all elements connected to the signal with the same name will have electric connection between them.

- **Power rails (VCC, GND)** are just a special type of signals, despite having special symbols assigned. The board can contain several power independent power rails, if the names are different.

Sensor board

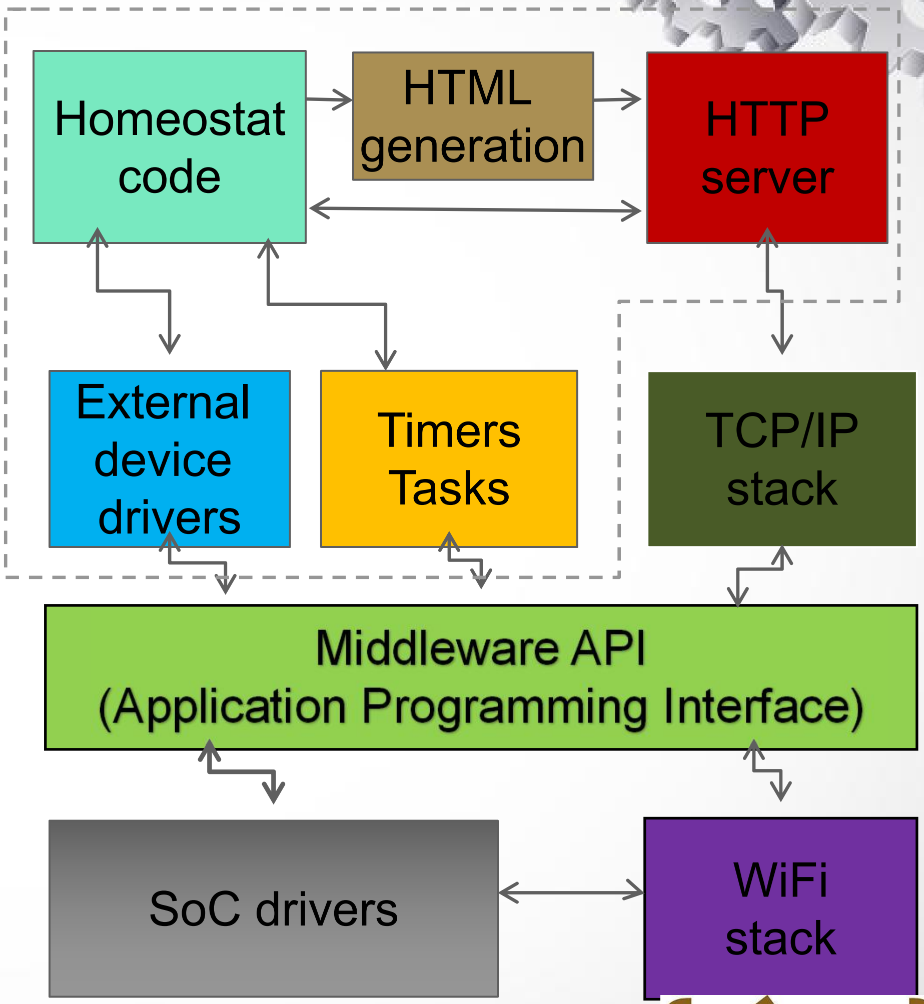


Gerber files creation

- Gerber - simple text format for 2D binary images.
- Every layer goes to its separate file:
  - .cmp : Component side (top)
  - .sol : Soldering side (bottom)
  - .plc : solk screen
- Drilling: Excellon format: drd
- Gerber viewers: gerbv or online

Homeostat - code concept

- The device measures the temperature and luminocity constantly
- The device creates the Wi-Fi Access point
- The user can connect to the Wi-Fi with his or her device
- The device services a simple web page with homeostatic controls (threshold level and event type)
- If the threshold is set, the device drives the output according to the selected rule and conditions



NodeMCU and Lua

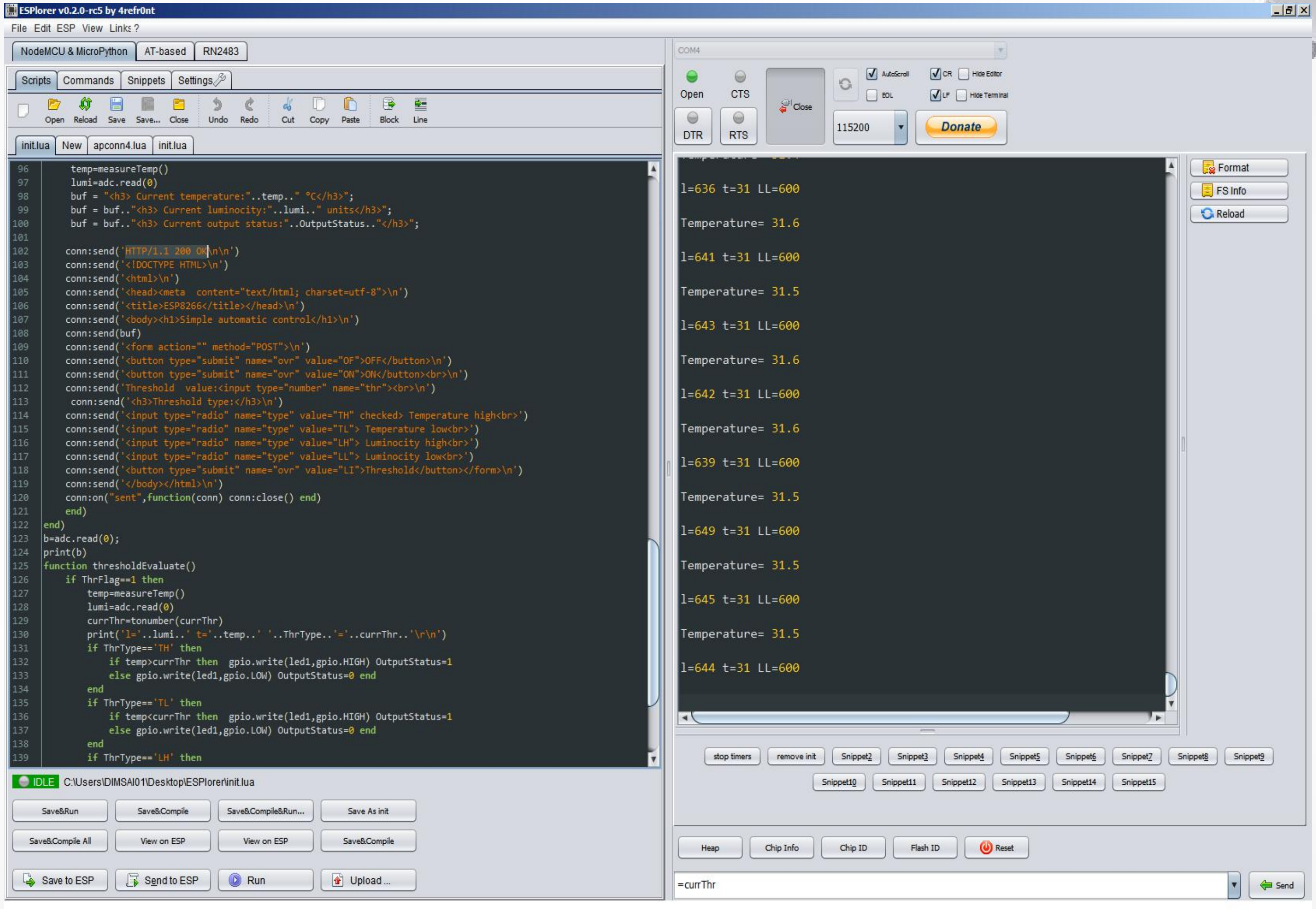
NodeMCU

- Open-source IoT platform for ESP8266
- Broad set of libraries and code snippets to use
- Pseudo-RTOS environment
- https://nodemcu.readthedocs.io/

Lua scripting language

- Lightweight interpretable language
- Implemented in C, shares the C-like language
- Multi-paradigm, rich set of native language tools
- https://en.wikipedia.org/wiki/Lua_(programming_language)

ESPlorer IDE



```lua
     temp=measureTemp()
     lumi=adc.read(0)
     buf = "<h3> Current temperature:"..temp.." °C</h3>";
     buf = buf.."<h3> Current luminocity:"..lumi.." units</h3>";
     buf = buf.."<h3> Current output status:"..OutputStatus.."</h3>";

    conn:send('HTTP/1.1 200 OK\n\n')
    conn:send('<!DOCTYPE HTML>\n')
    conn:send('<html>\n')
    conn:send('<head><meta  content="text/html; charset=utf-8">\n')
    conn:send('<title>ESP8266</title></head>\n')
    conn:send('<body><h1>Simple automatic control</h1>\n')
    conn:send(buf)
    conn:send('<form action="" method="POST">\n')
    conn:send('<button type="submit" name="ovr" value="OF">OFF</button>\n')
    conn:send('<button type="submit" name="ovr" value="ON">ON</button><br>\n')
    conn:send('Threshold  value:<input type="number" name="thr"><br>\n')
     conn:send('<h3>Threshold type:</h3>\n')
    conn:send('<input type="radio" name="type" value="TH" checked> Temperature high<br>')
    conn:send('<input type="radio" name="type" value="TL"> Temperature low<br>')
    conn:send('<input type="radio" name="type" value="LH"> Luminocity high<br>')
    conn:send('<input type="radio" name="type" value="LL"> Luminocity low<br>')
    conn:send('<button type="submit" name="ovr" value="LI">Threshold</button></form>\n')
    conn:send('</body></html>\n')
    conn:on("sent",function(conn) conn:close() end)
    end)
end)
b=adc.read(0);
print(b)
function thresholdEvaluate()
    if ThrFlag==1 then
        temp=measureTemp()
        lumi=adc.read(0)
        currThr=tonumber(currThr)
        print('l='..lumi..' t='..temp..' '..ThrType..'='..currThr..'\r\n')
        if ThrType=='TH' then
            if temp>currThr then  gpio.write(led1,gpio.HIGH) OutputStatus=1
            else gpio.write(led1,gpio.LOW) OutputStatus=0 end
        end
        if ThrType=='TL' then
            if temp<currThr then  gpio.write(led1,gpio.HIGH) OutputStatus=1
            else gpio.write(led1,gpio.LOW) OutputStatus=0 end
        end
        if ThrType=='LH' then
```

C:\Users\DIMSAI01\Desktop\ESPlorer\init.lua

```
l=636 t=31 LL=600

Temperature= 31.6

l=641 t=31 LL=600

Temperature= 31.5

l=643 t=31 LL=600

Temperature= 31.6

l=642 t=31 LL=600

Temperature= 31.6

l=639 t=31 LL=600

Temperature= 31.5

l=649 t=31 LL=600

Temperature= 31.5

l=645 t=31 LL=600

Temperature= 31.5

l=644 t=31 LL=600
```

```
=currThr
```

The code elements

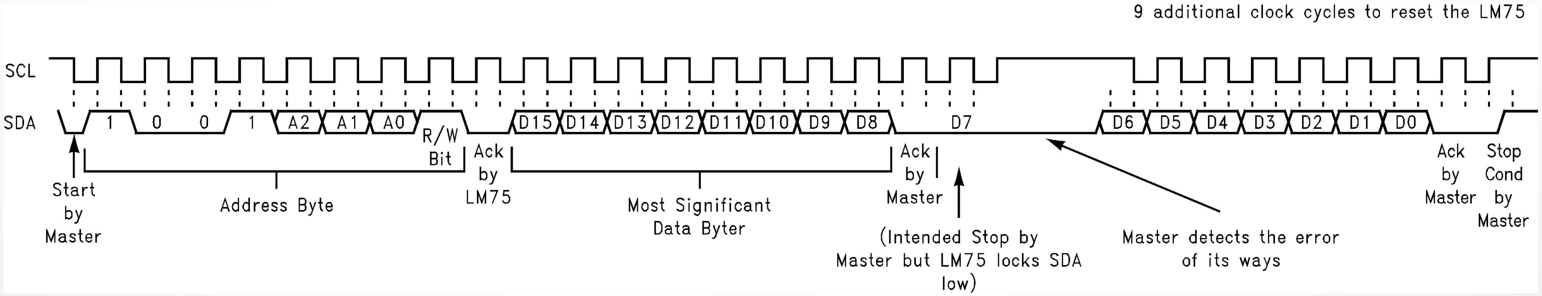
- Wi-Fi initialization
- DHCP initialization
- Thermometer communication library (I2C)
- A simple web server (serving the page)
- The page contents (HTML)
- POST response parser
- Threshold checker and timer service routine

Wi-Fi and DHCP

- The homeostat will create a tiny access point (AP): it creates it's own network, provides the low-level communication protocol and authorization.
- Dynamic Host Configuration Protocol - an “address book” service in the network. Assigns the addresses for the connected devices and manages their configuration.
- Good news: they are provided by NodeMCU!

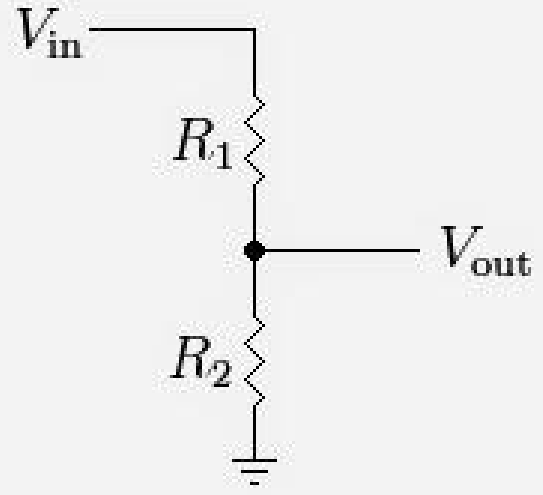
Reading out the thermometer

```
i2c.setup(0, sda, scl, i2c.SLOW)

function read_reg(dev_addr,bytes)
   i2c.start(0)
   i2c.start(0)
   i2c.address(0, dev_addr, i2c.RECEIVER)
   c = i2c.read(0, bytes) -- bytes read and returned
   i2c.stop(0)
   return c
end
function measureTemp()
   reg = read_reg(LM75_addr,2) --for brd#3

temp=10*tonumber(string.byte(reg,1))+(tonumber(string.byte(reg,2))
/32)
   print(string.format("Temperature= %d.%d \r\n",temp/10,temp-
(temp/10)*10))
   return temp/10
end
```


9 additional clock cycles to reset the LM75
SCL
SDA
1
0
0
1
A2
A1
A0
R/W Bit
D15
D14
D13
D12
D11
D10
D9
D8
D7
D6
D5
D4
D3
D2
D1
D0
Start by Master
Address Byte
Ack by LM75
Most Significant Data Byter
Ack by Master
(Intended Stop by Master but LM75 locks SDA low)
Master detects the error of its ways
Ack by Master
Stop Cond by Master

Reading out the photo resistor

- ESP8266 has very basic 8-bit analog-digital converter that can convert the voltage in the range of 0..1.024V
- Our photoresistor range is 0.45..16 kOhm
- Out input voltage is 3.3 V
- API command: adc.read(0)

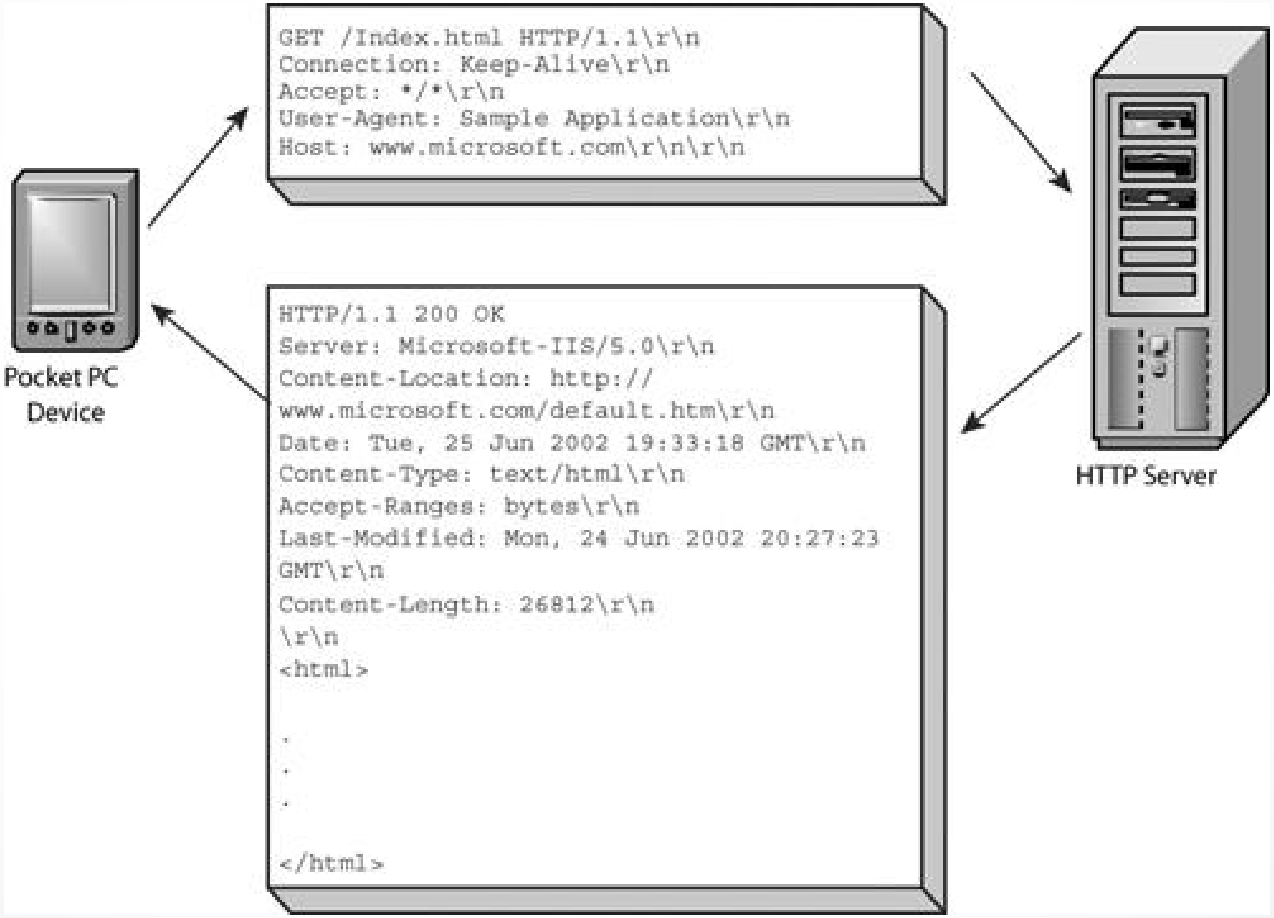


$$V_{out} = \frac{R_2}{R_1 + R_2} \cdot V_{in}$$

HTTP protocol basics

- A stateless simple “request-response” protocol.
- Computers are communicating with plain text.
- The client asks for the specific page on the server and describes what type of content it can accept.
- The server replies with the basic information about the content provided and serves the page itself.

HTTP protocol: example



```
GET /Index.html HTTP/1.1\r\n
Connection: Keep-Alive\r\n
Accept: */*\r\n
User-Agent: Sample Application\r\n
Host: www.microsoft.com\r\n\r\n
```

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0\r\n
Content-Location: http://
www.microsoft.com/default.htm\r\n
Date: Tue, 25 Jun 2002 19:33:18 GMT\r\n
Content-Type: text/html\r\n
Accept-Ranges: bytes\r\n
Last-Modified: Mon, 24 Jun 2002 20:27:23
GMT\r\n
Content-Length: 26812\r\n
\r\n
<html>


.
.
.

</html>
```

HTTP POST method

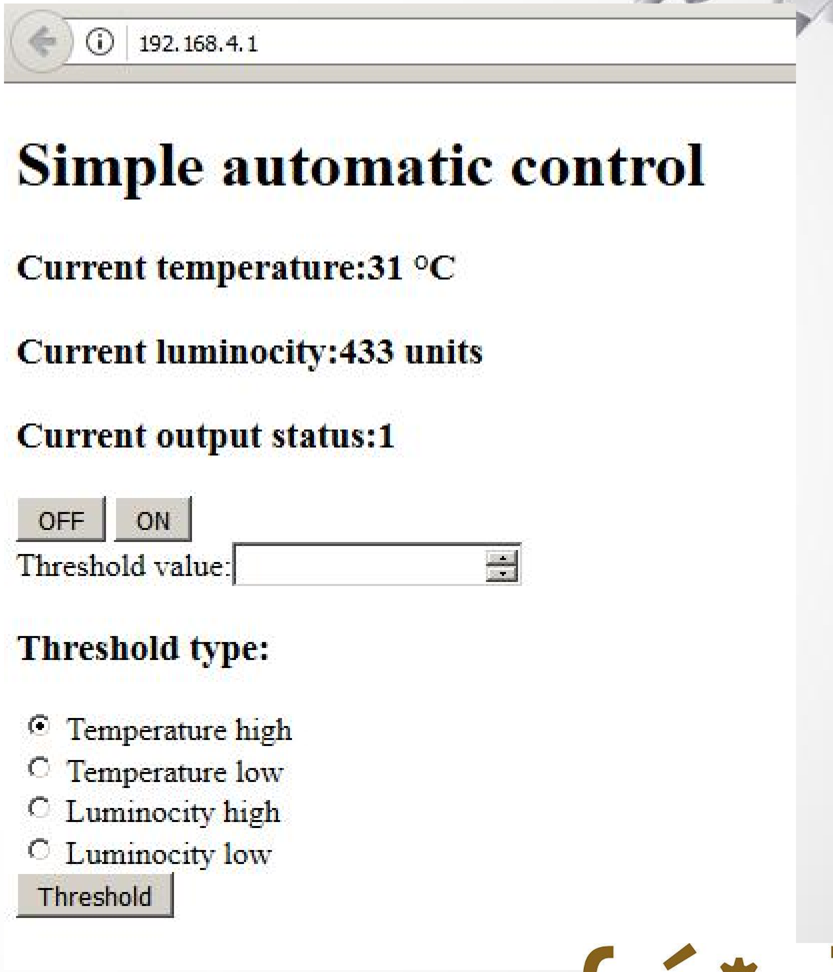
We use POST requests to submit the data to the server.

The server gets the string of parameter-value pairs, encoded as: parameter1=value1&parameter2=value2 &parameter3=value3 etc.

All values are plain text strings

The values go in arbitrary order

We need to transmit three parameters:
- Action (On, Off, Set threshold)
- Threshold type
- Threshold parameter value



192.168.4.1

Simple automatic control

Current temperature:31 °C

Current luminocity:433 units

Current output status:1

Threshold value:

Threshold type:

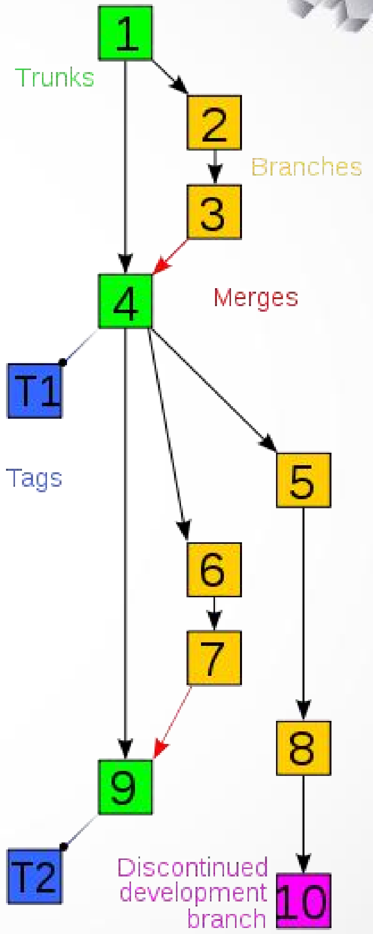
● Temperature high
○ Temperature low
○ Luminocity high
○ Luminocity low

HTML code for our web page

```
<!DOCTYPE HTML>
<html>
<head><meta  content="text/html; charset=utf-8">
<title>ESP8266</title></head>
<body><h1>Simple automatic control</h1>
<h3> Current temperature:"..temp.." °C</h3>
<h3> Current luminocity:"..lumi.." units</h3>
<h3> Current output status:"..OutputStatus.."</h3>
<form action="" method="POST">
<button type="submit" name="ovr" value="OF">OFF</button>
<button type="submit" name="ovr" value="ON">ON</button><br>
Threshold  value:<input type="number" name="thr"><br>
 <h3>Threshold type:</h3>
<input type="radio" name="type" value="TH" checked> Temperature high<br>
<input type="radio" name="type" value="TL"> Temperature low<br>
<input type="radio" name="type" value="LH"> Luminocity high<br>
<input type="radio" name="type" value="LL"> Luminocity low<br>
<button type="submit" name="ovr" value="LI">Threshold</button></form>
</body></html>
```

Project publication

- VCS - version control software, a de-facto standard of project management in 2017
- Most popular systems: git, Subversion, Mercurial
- The biggest repository of opensource projects: Github



Project publication

We need to deliver:

- Design files
- Code
- Firmware version (or the firmware itself)
- Assembly instructions
- Security note: avoid leaving sensitive data in your code! (like passwords, IP addresses etc)