PERVASIVE SYSTEMS 2015/2016 SAPIENZA DIAG

HA, SMART HOME & ARDUINO

20 APRIL 2016

Massimo Perri

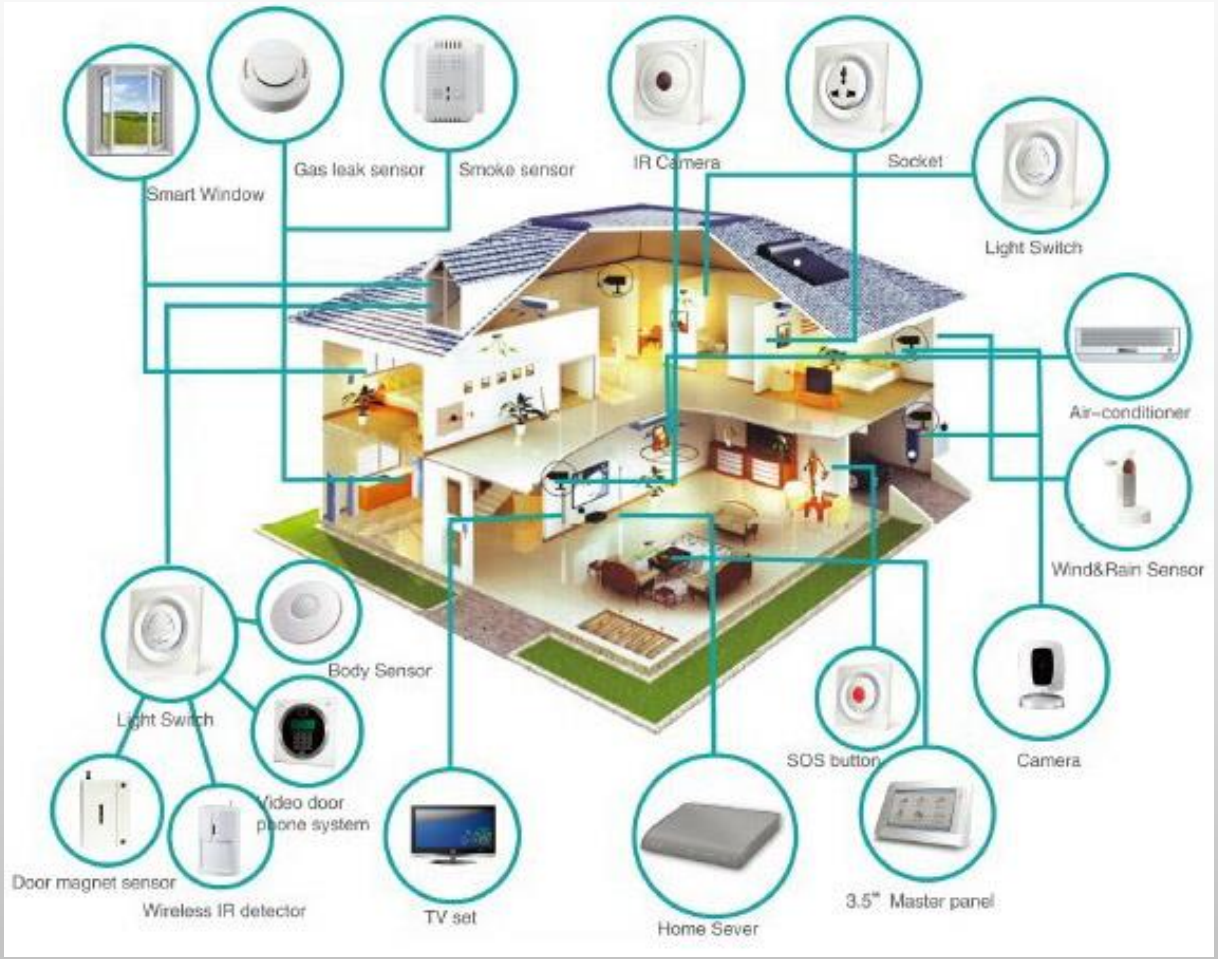
perri.1632744@studenti.uniroma1.it

https://github.com/mp-76/smarthome_arduino

What is a Smart Home ?

- UK Department of Trade and Industry definition: "A dwelling incorporating a communications network that connects the key electrical appliances and services, and allows them to be remotely controlled, monitored or accessed.”
- Smart Homes (and in general Home Automation) target is to improve people’s quality of life and time they spend at home

Smart Home: what ? how ?



- Lighting
- Heating/ventilation & Air Conditioning
- Shutter/Blind & shading
- Alarm monitoring
- Energy management & metering
- Audio & video distribution

How ? Address each device on a bus and remotely control it with messages.

HA/Smart Home Standards

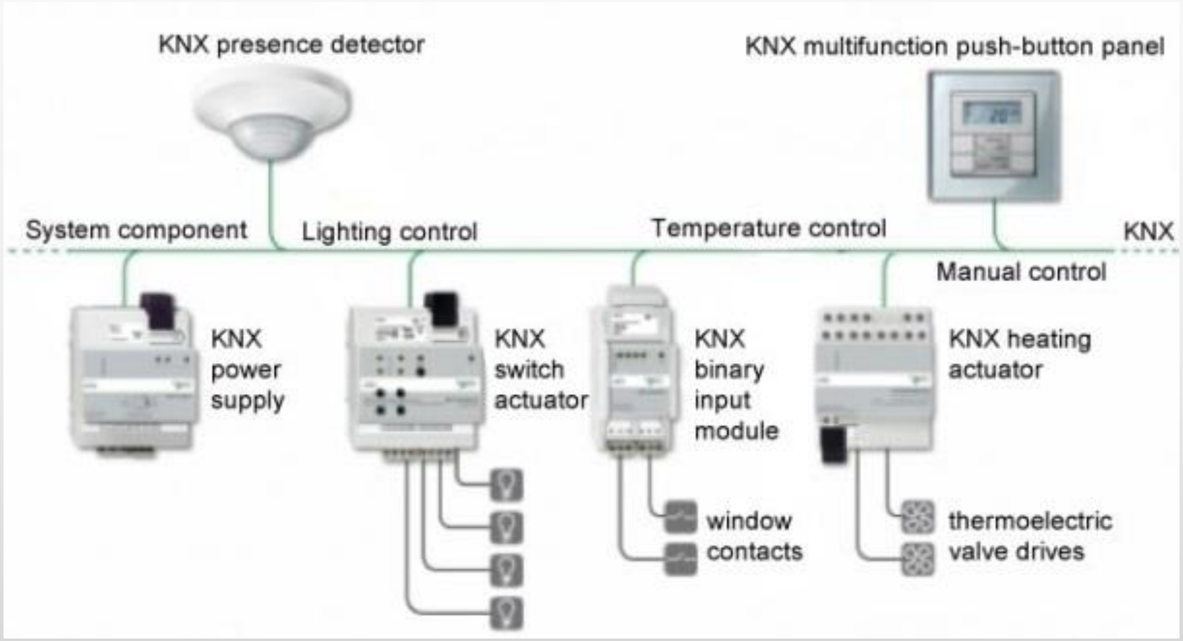
- X10
- Insteon
- ZigBee
- Z-Wave
- CBUS
- bTicino ‘MyHome’
- KNX

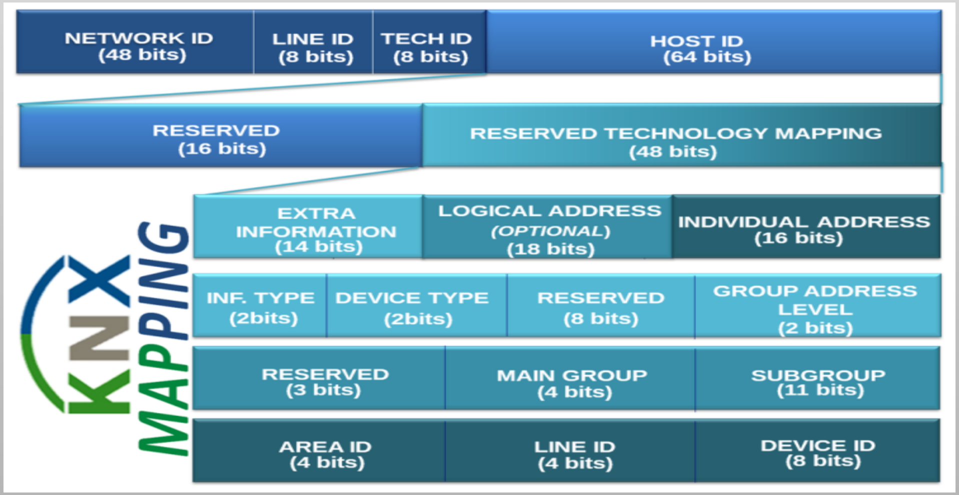
https://www.knx.org/knx-en/index.php

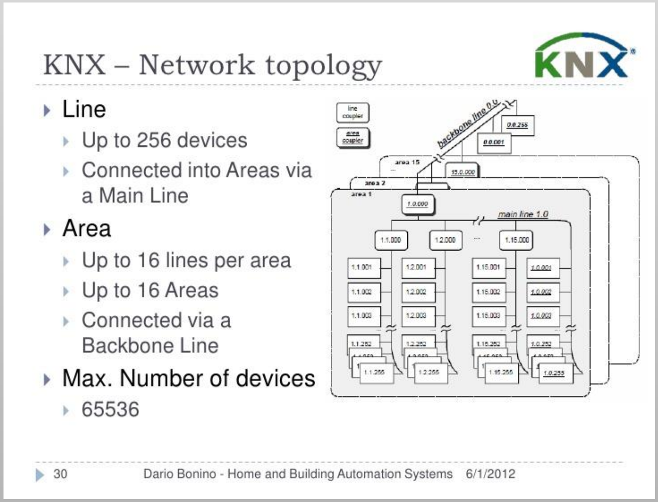
KNX : powerful, open, free

- KNX uses a dedicated 2-wires control bus
- can be programmed with a PC by USB/Ethernet to the bus
- devices are produced by ~400 different manifacturers in 37 countries
- powerful : discovery + group addressing; addressing space up to 65536 devices

KNX: topologies, addressing, controls



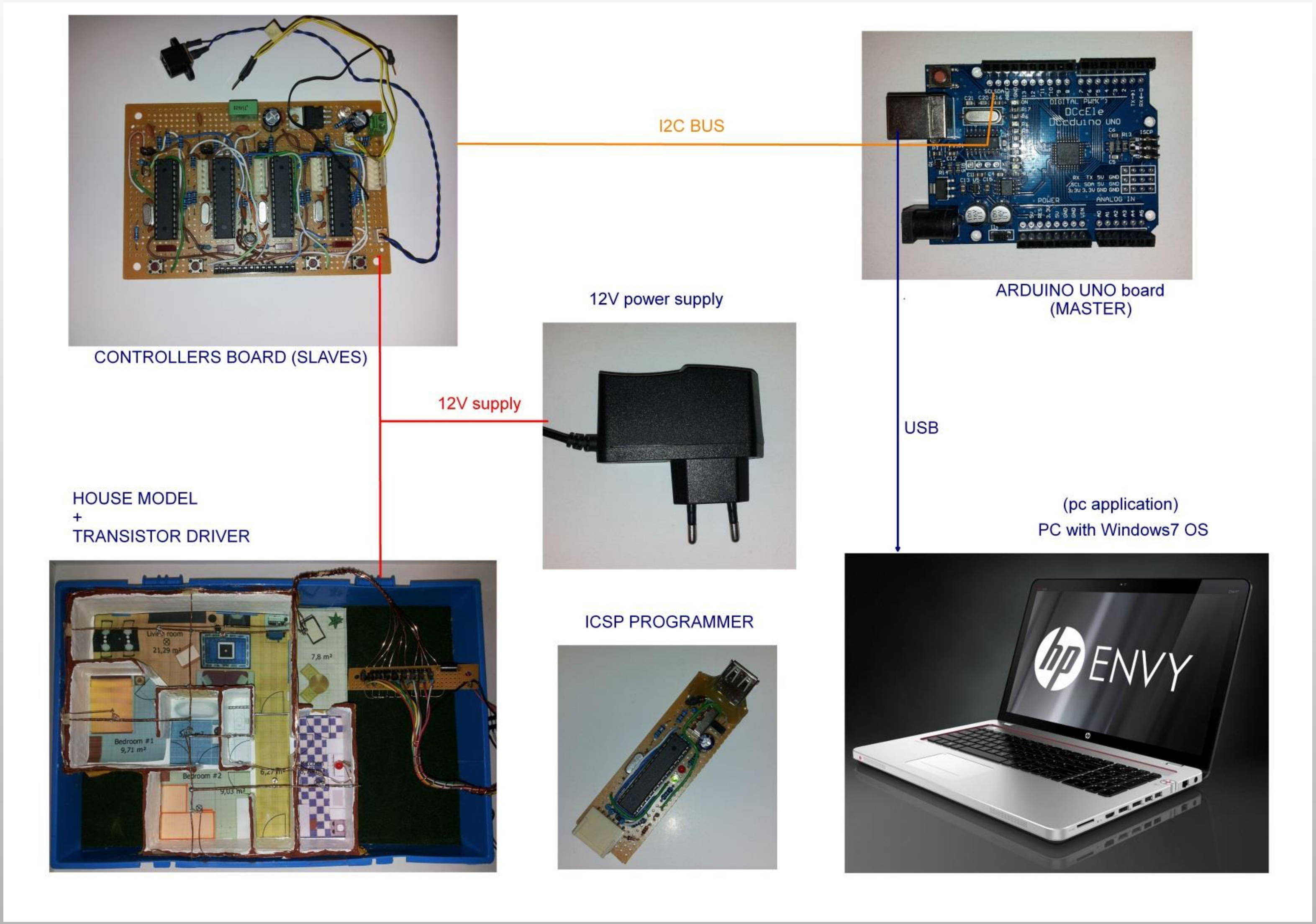



KNX – Network topology

- Line
  - Up to 256 devices
  - Connected into Areas via a Main Line
- Area
  - Up to 16 lines per area
  - Up to 16 Areas
  - Connected via a Backbone Line
- Max. Number of devices
  - 65536

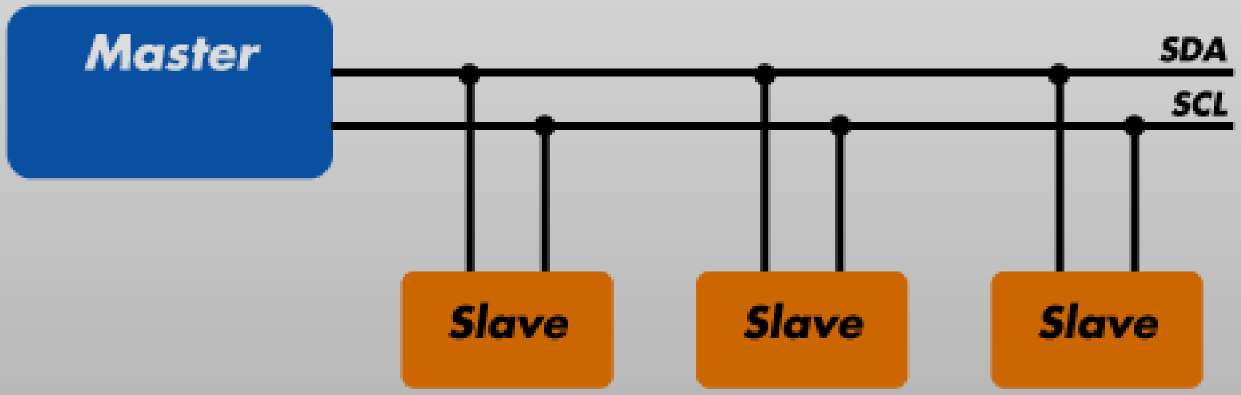
30 Dario Bonino - Home and Building Automation Systems 6/1/2012

A simple HA Arduino based demo system inspired by KNX (1/2)



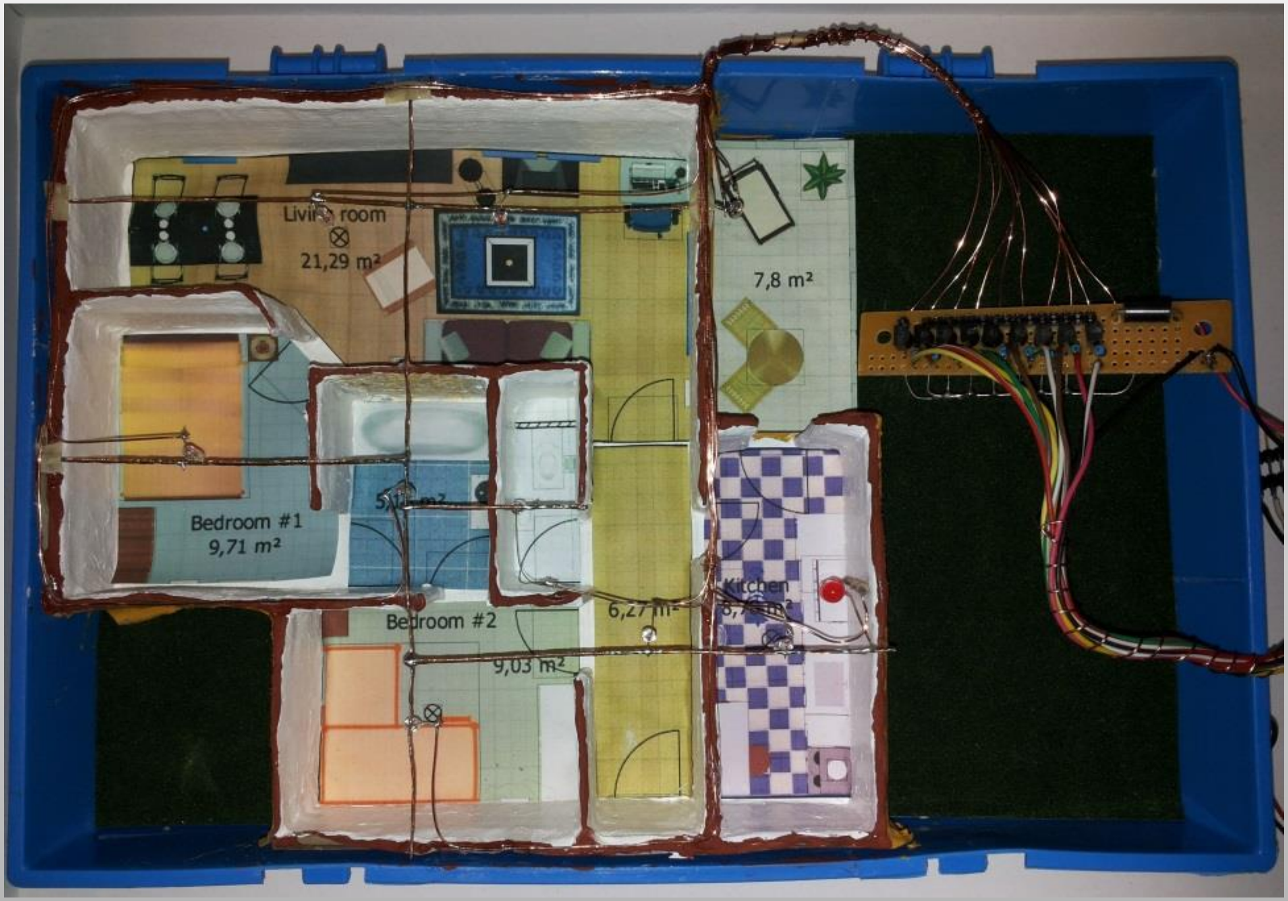
A simple HA Arduino based demo system inspired by KNX (2/2)

- Demo: simple control bus + user interface as PC application + Arduino & some ATMEGA328
- PC application discovers devices connected on bus
- Arduino I2C master initially unaware of controllers present on the bus and devices → **discovery like in KNX**
- 4x Controllers independent from one each, they host devices, actuators or sensors → **logically addressable like in KNX**

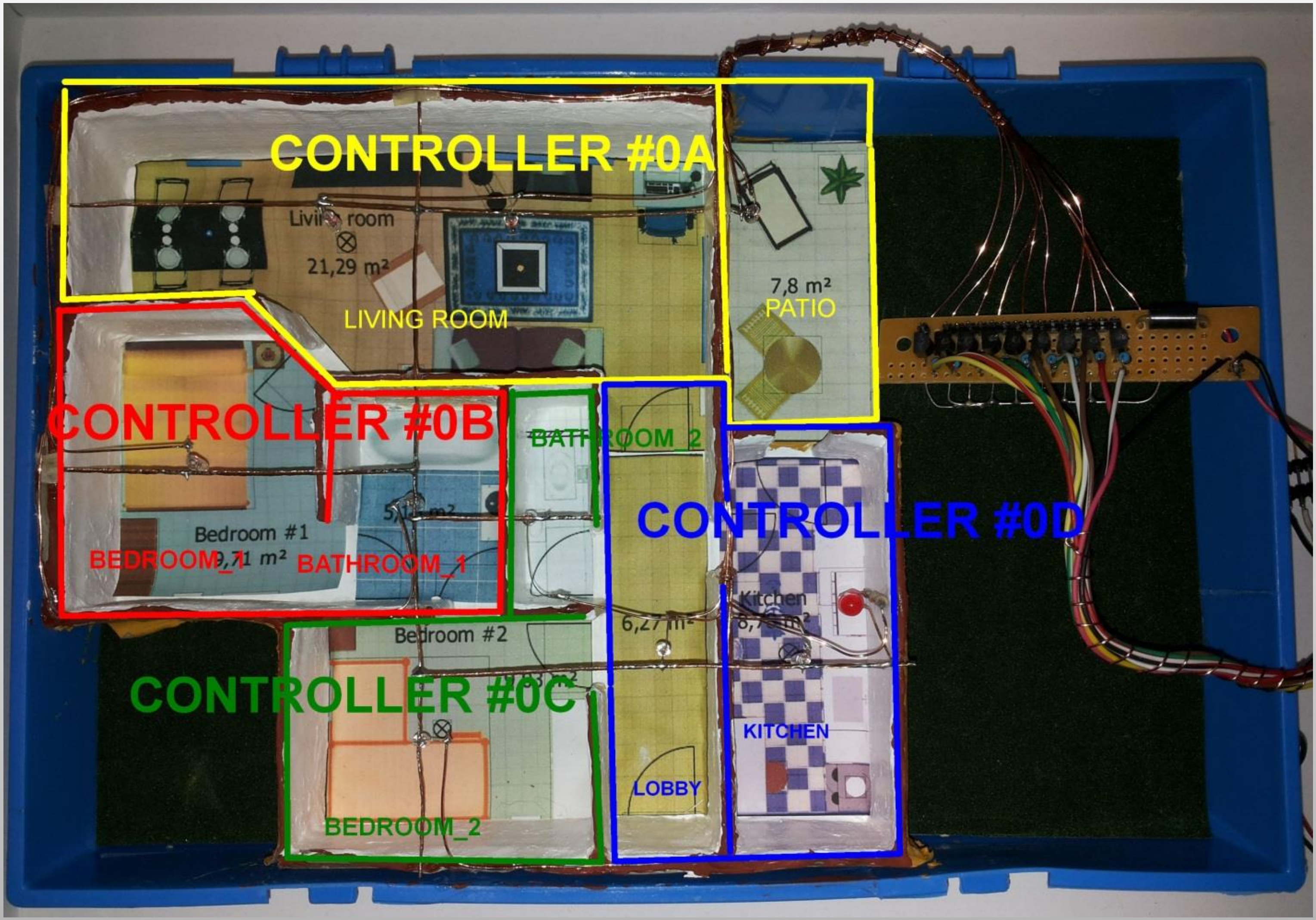


House model and transistors driver

- The controllers/slaves are connected to a scaled house model
- Actuators (lamps and LED) are of two kinds , LPWM and LDIG
- LPWM can be dimmed (8 bits, values 0-255)
- LDIG can be set ON or OFF

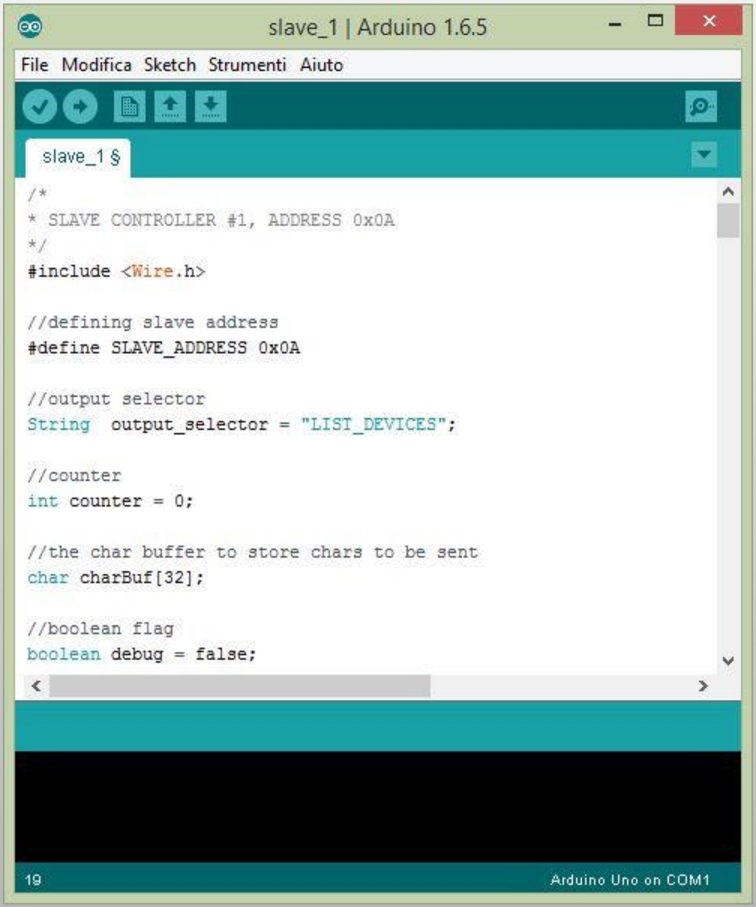


House model : controllers areas and mappings

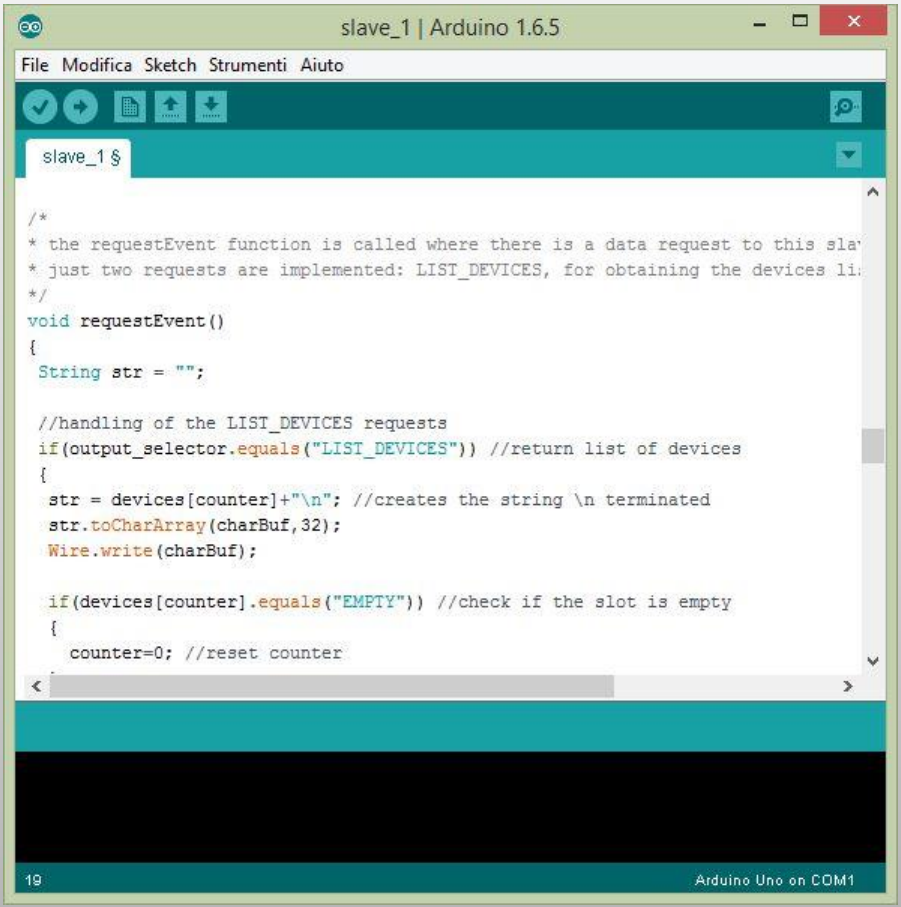


Controllers/Slaves programs

- Addressing of devices for I2C
- Event driven code (empty main loop)
- I2C protocol to the arduino master
- Functions/commands implemented: LIST_DEVICES, SET , READ



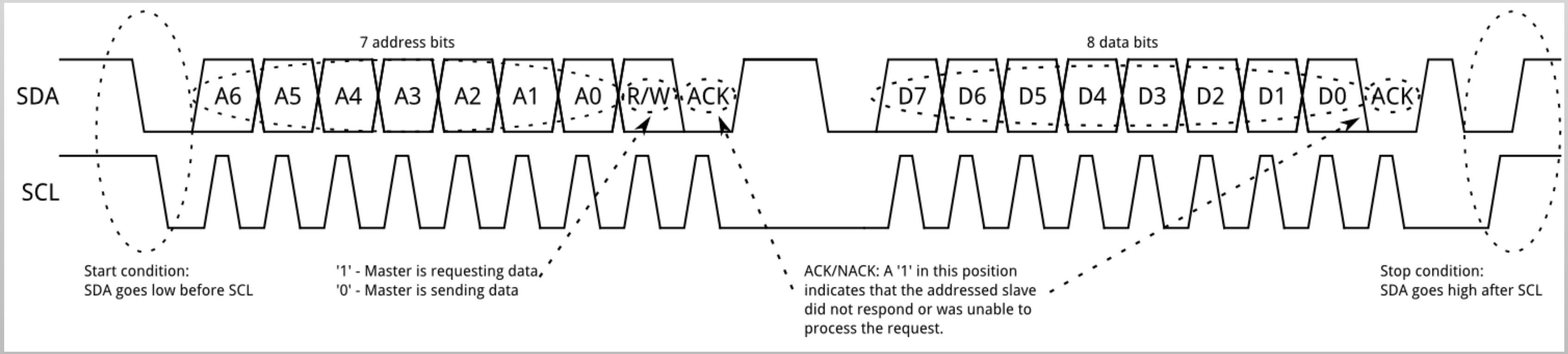
```
/*
* SLAVE CONTROLLER #1, ADDRESS 0x0A
*/
#include <Wire.h>

//defining slave address
#define SLAVE_ADDRESS 0x0A

//output selector
String  output_selector = "LIST_DEVICES";

//counter
int counter = 0;

//the char buffer to store chars to be sent
char charBuf[32];

//boolean flag
boolean debug = false;
```



```
/*
* the requestEvent function is called where there is a data request to this sla
* just two requests are implemented: LIST_DEVICES, for obtaining the devices li
*/
void requestEvent()
{
 String str = "";

 //handling of the LIST_DEVICES requests
 if(output_selector.equals("LIST_DEVICES")) //return list of devices
 {
  str = devices[counter]+"\n"; //creates the string \n terminated
  str.toCharArray(charBuf,32);
  Wire.write(charBuf);

  if(devices[counter].equals("EMPTY")) //check if the slot is empty
  {
    counter=0; //reset counter
```

Discovery & controls

- A scanner procedure is initiated to discover controllers and their associated devices
- At the Arduino master a local data structure is built
- Controllers (ATMEGA328P) reply to the master and execute commands to set actuators and read from sensors



KNX recall and mirroring

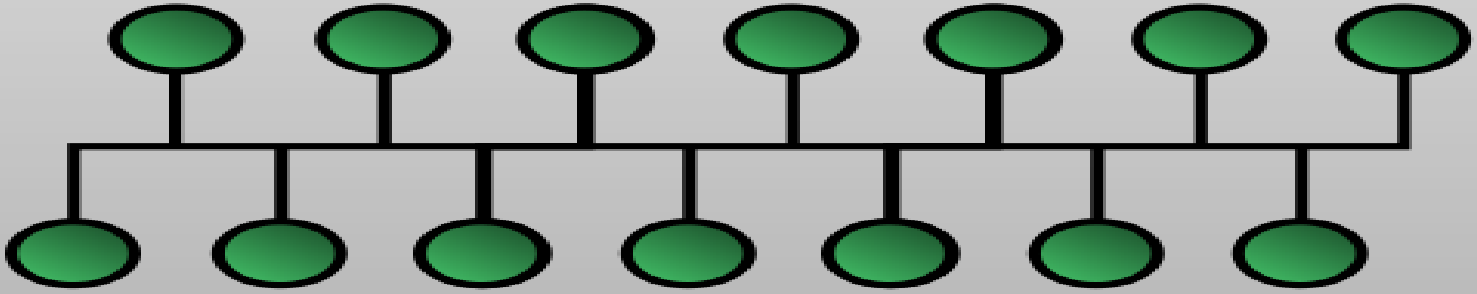
- The system is inspired by KNX , its addressing and 2-wires bus schema
- KNX addressing is made this way: AREA.LINE.DEVICE e.g. 2.1.1
- Use hi part of controller address for AREA, lower part for LINE and devices ordering number as DEVICE is logical to the controller:

example, if we have a controller with address 0x21 with a LPWM device with device_id = 1 connected we got a virtual address of 2.1.1 for the LPWM device

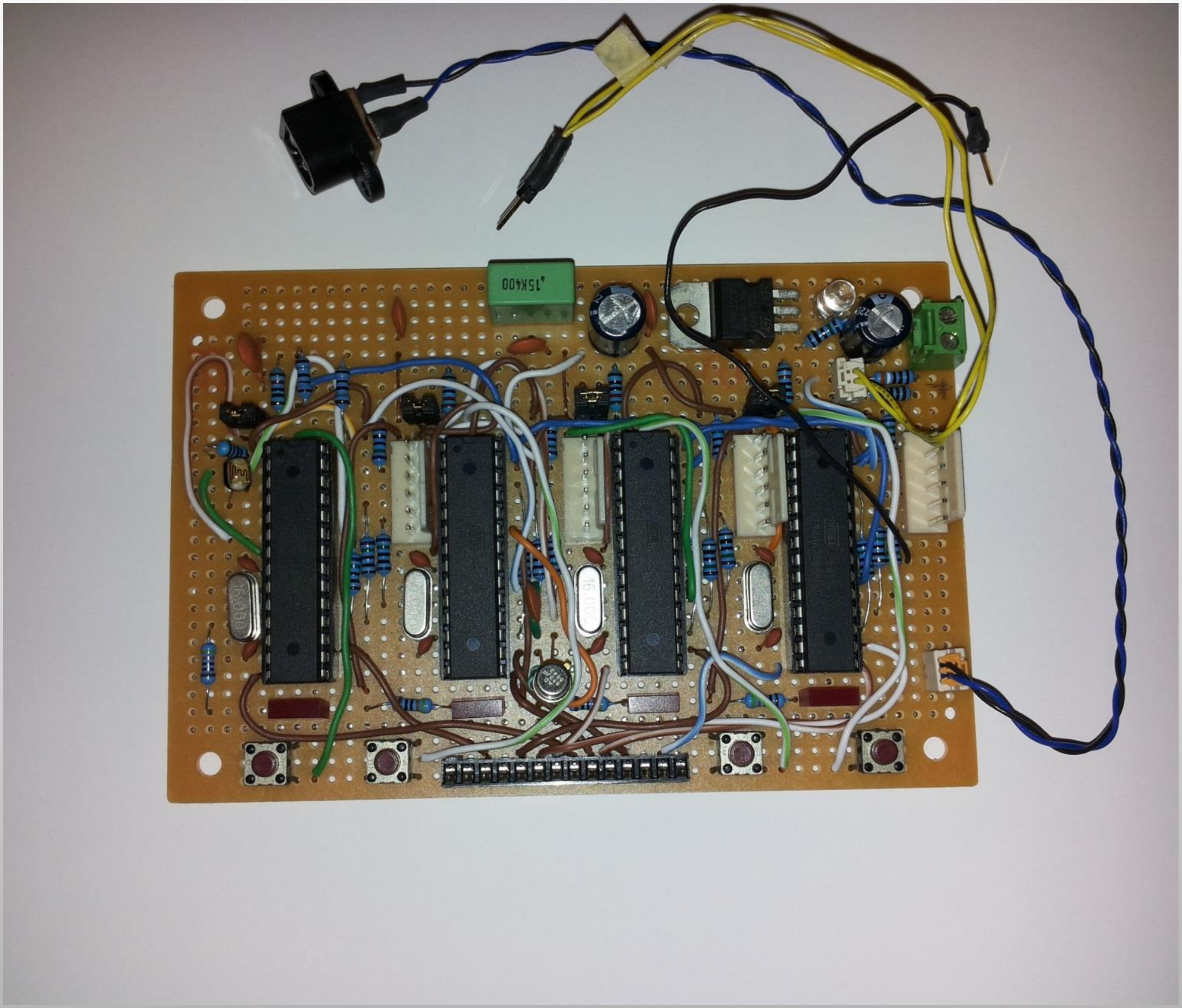
I2C limitations and constraints

- I2C requires the presence of a master
- There is the need of pulling-up SCL and SDA lines
- The addressing scheme use 7-bits : just 112 allowed
- The idea is to split the I2C address in AREA and LINE parts (see KNX) + an inner logical address for devices (DEVICE)

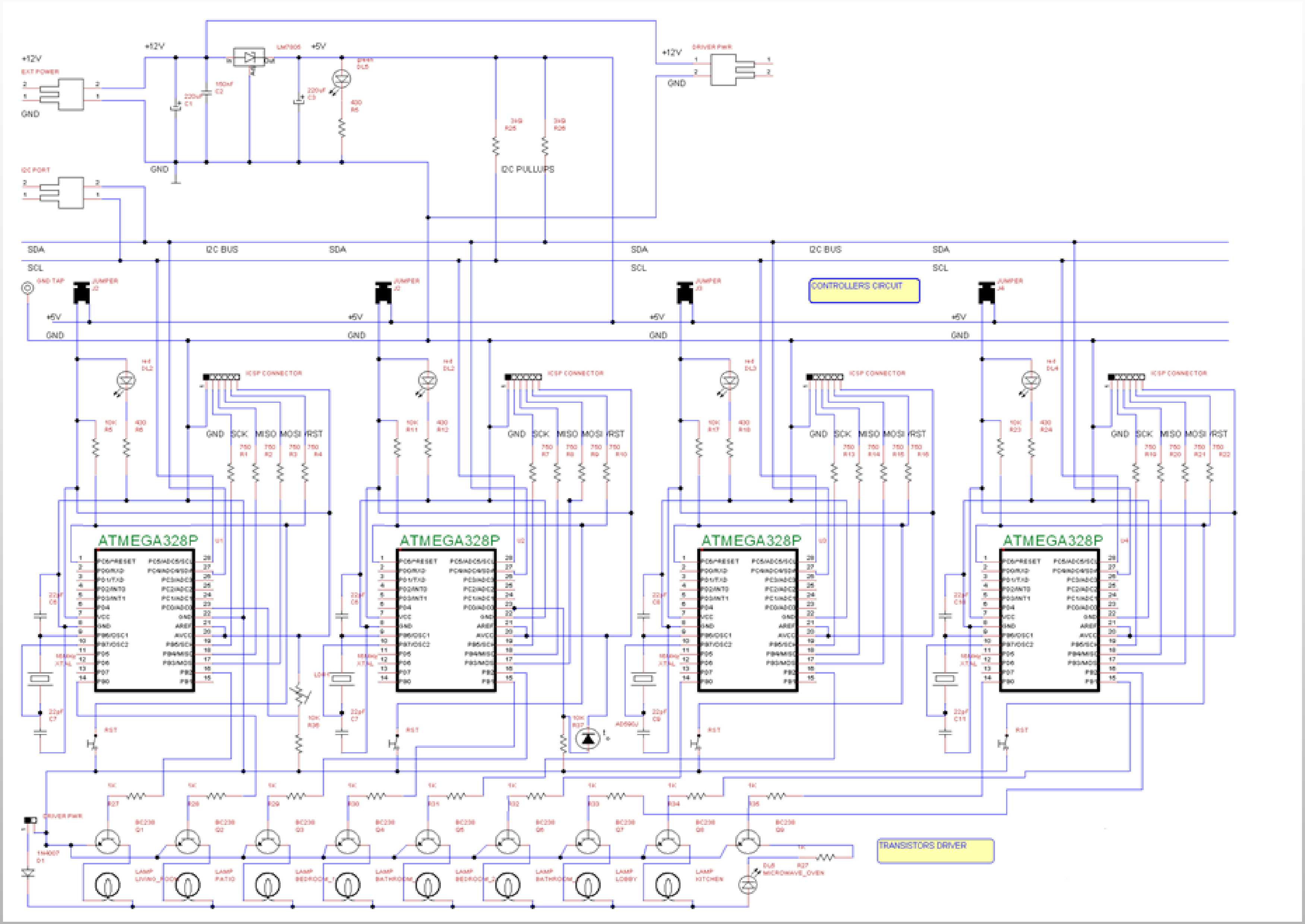
We may have 10 areas, 10 lines per area and 10 devices per line → **for a total of 1000 distinct devices logically addressable on the bus**



Controllers Board



Controllers board + driver schematics

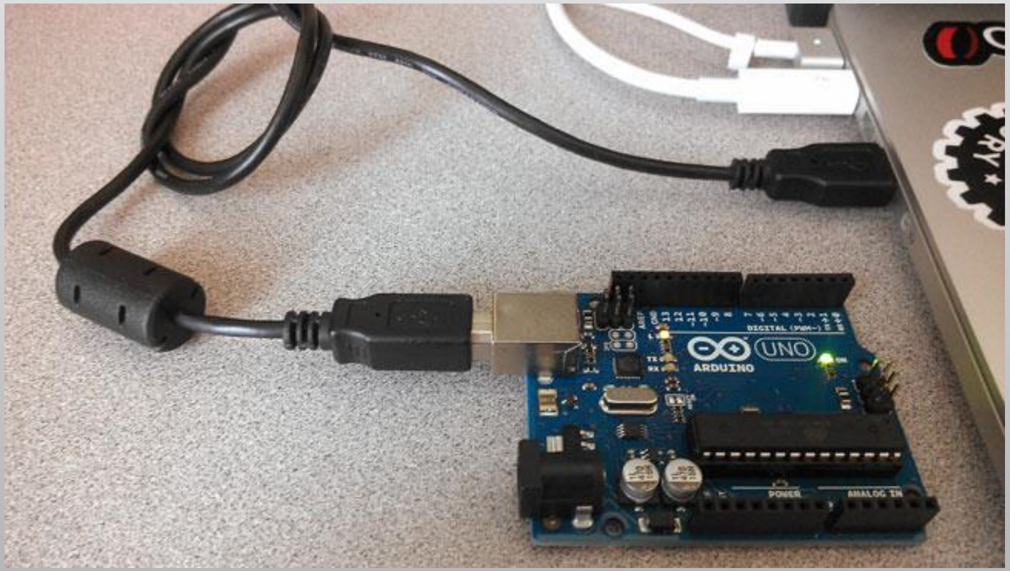


PC Application : pc-master protocol

PC app issues ‘CMD SCAN BUS’ to the Arduino master using a function:

```
public House scanBusForModel()
{
  sendSerialMsg(port, "CMD SCAN BUS");
  String str = awaitSerialMsgUntil(port, '-');

  return loadHouseModel(str);
}
```

- CMD SET <actuator> <value>
- CMD READ <sensor>



The master-slave protocol

- Send of a command

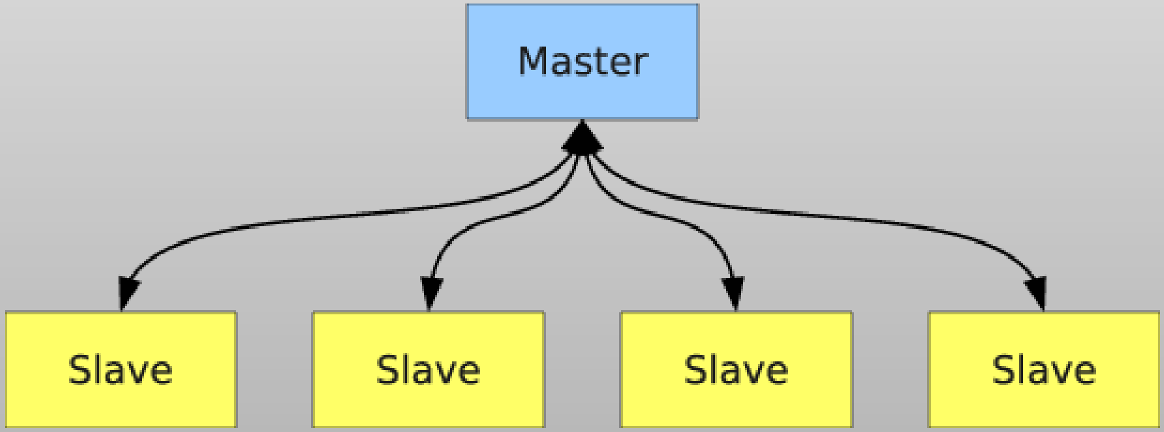
  sendCommand(slave_address, "CMD LIST_DEVICES");

- Read of a sensor

  String readSensor(int slave_address,String sensor_name, String sensor_type)

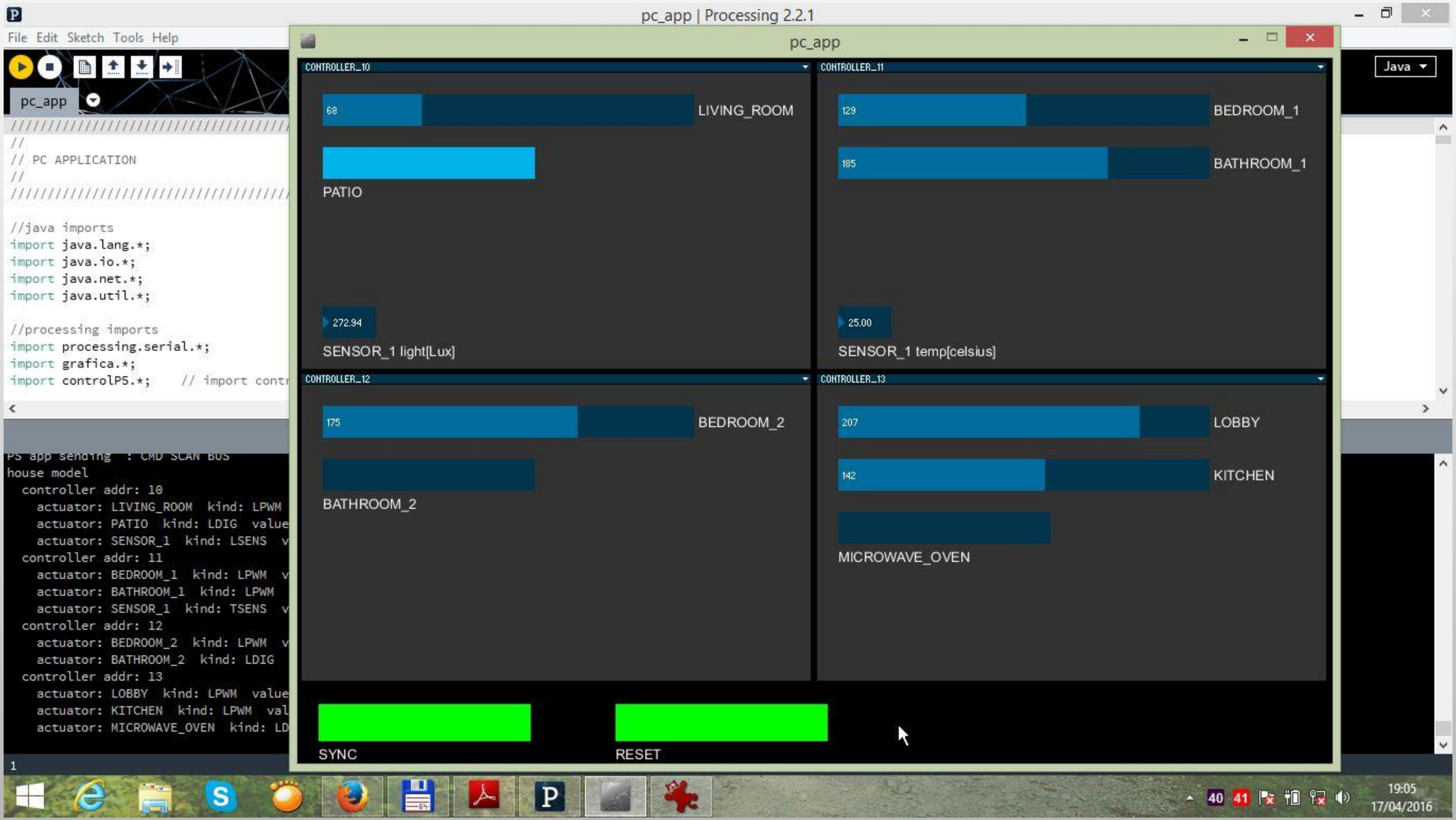
- Set of an actuator

  void setActuator(int slave_address, String actuator_name, int value)



PC Application : Processing and Java

- Easy dinamic interface , unaware of controllers and devices until first sync
- https://github.com/mp-76/smarthome_arduino



```
/////////////////////////////////////
//
// PC APPLICATION
//
/////////////////////////////////////

//java imports
import java.lang.*;
import java.io.*;
import java.net.*;
import java.util.*;

//processing imports
import processing.serial.*;
import grafica.*;
import controlP5.*;     // import contr
```

```
PS app sending  : CMD SCAN BUS
house model
  controller addr: 10
    actuator: LIVING_ROOM  kind: LPWM
    actuator: PATIO  kind: LDIG  value
    actuator: SENSOR_1  kind: LSENS  v
  controller addr: 11
    actuator: BEDROOM_1  kind: LPWM  v
    actuator: BATHROOM_1  kind: LPWM
    actuator: SENSOR_1  kind: TSENS  v
  controller addr: 12
    actuator: BEDROOM_2  kind: LPWM  v
    actuator: BATHROOM_2  kind: LDIG
  controller addr: 13
    actuator: LOBBY  kind: LPWM  value
    actuator: KITCHEN  kind: LPWM  val
    actuator: MICROWAVE_OVEN  kind: LD
```
1

Thank you