Networking and communication

FabAcademy 2019

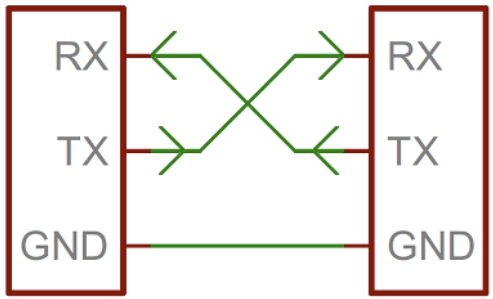
Wired Communication

- Asynchronous
- SPI
- I2C

Asynchronous Serial

It is the protocol also used by the mcu (attiny45, atmega….) when you print some information to the Serial Monitor.

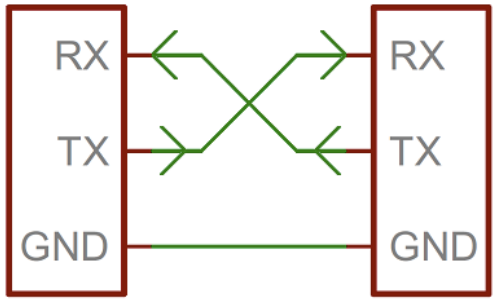
Asynchronous means that data is transferred **without support from an external clock signal.**



No Clock => Rules

Rules:
- Data bits,
- Synchronization bits,
- Parity bits
- Baud rate.

Asynchronous Serial



No Clock => Rules

Rules:
- Data bits,
- Synchronization bits,
- Parity bits
- Baud rate.

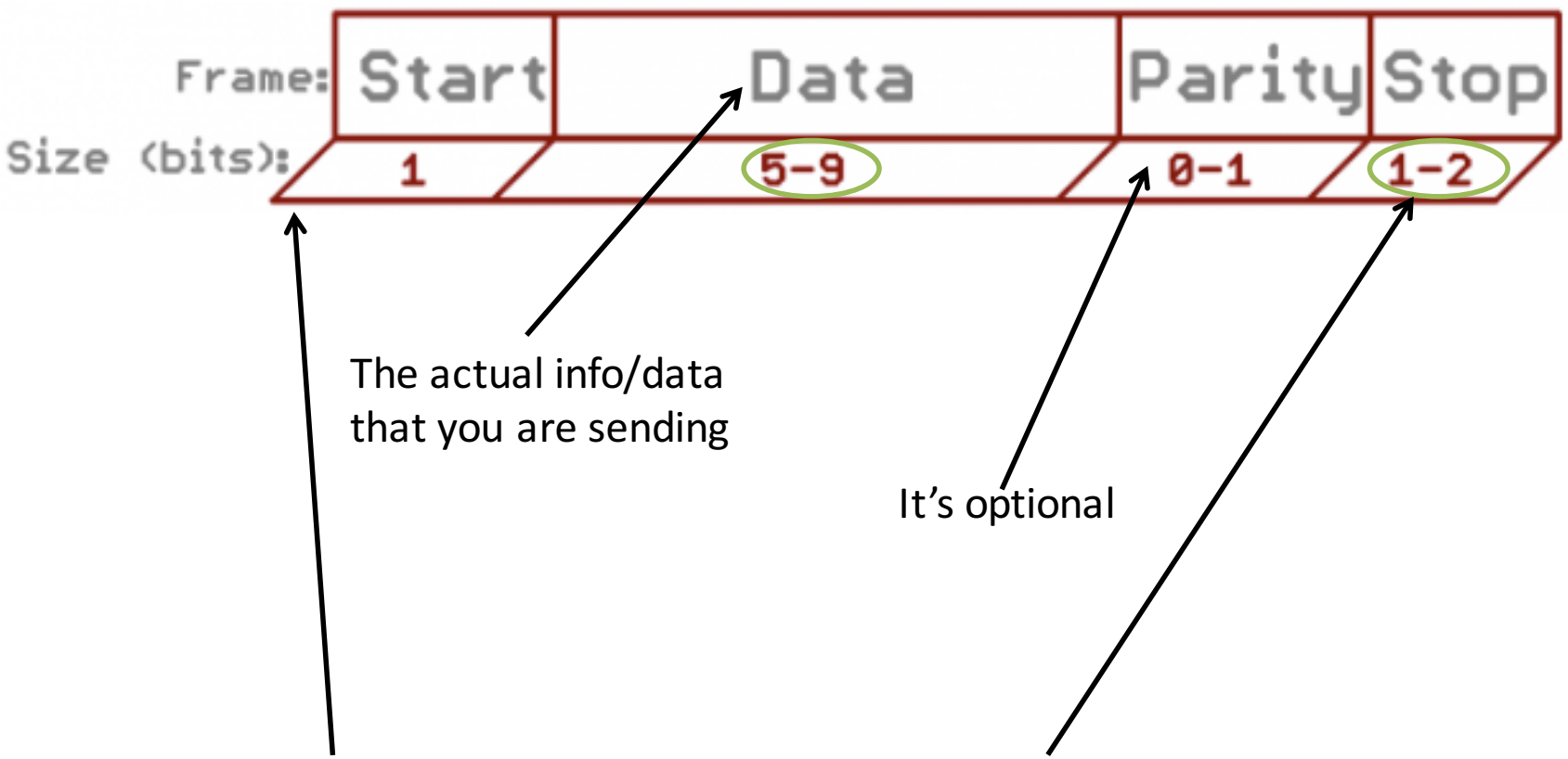
Baud Rate

How fast data is sent over a serial line: bps (bits-per-second).

“standard” baud:
1200, 2400, 4800, 19200, 38400, 57600,115200

In term of time?
9600 bps -> 1/9600 = 104us
115200 bps -> 1/115200 = 8.7us

Asynchronous Serial Packet and Bits



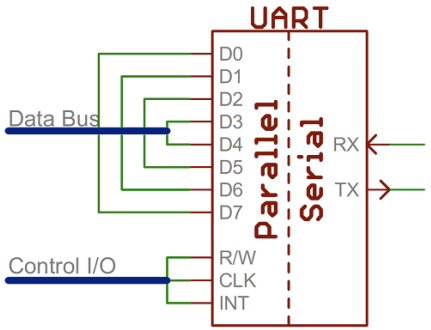
Asynchronous Serial Packet and Bits

EXAMPLE

Asynchronous Serial Hardware - Software

Hardware

A universal asynchronous receiver/transmitter (UART) is a block of circuitry responsible for implementing serial communication.



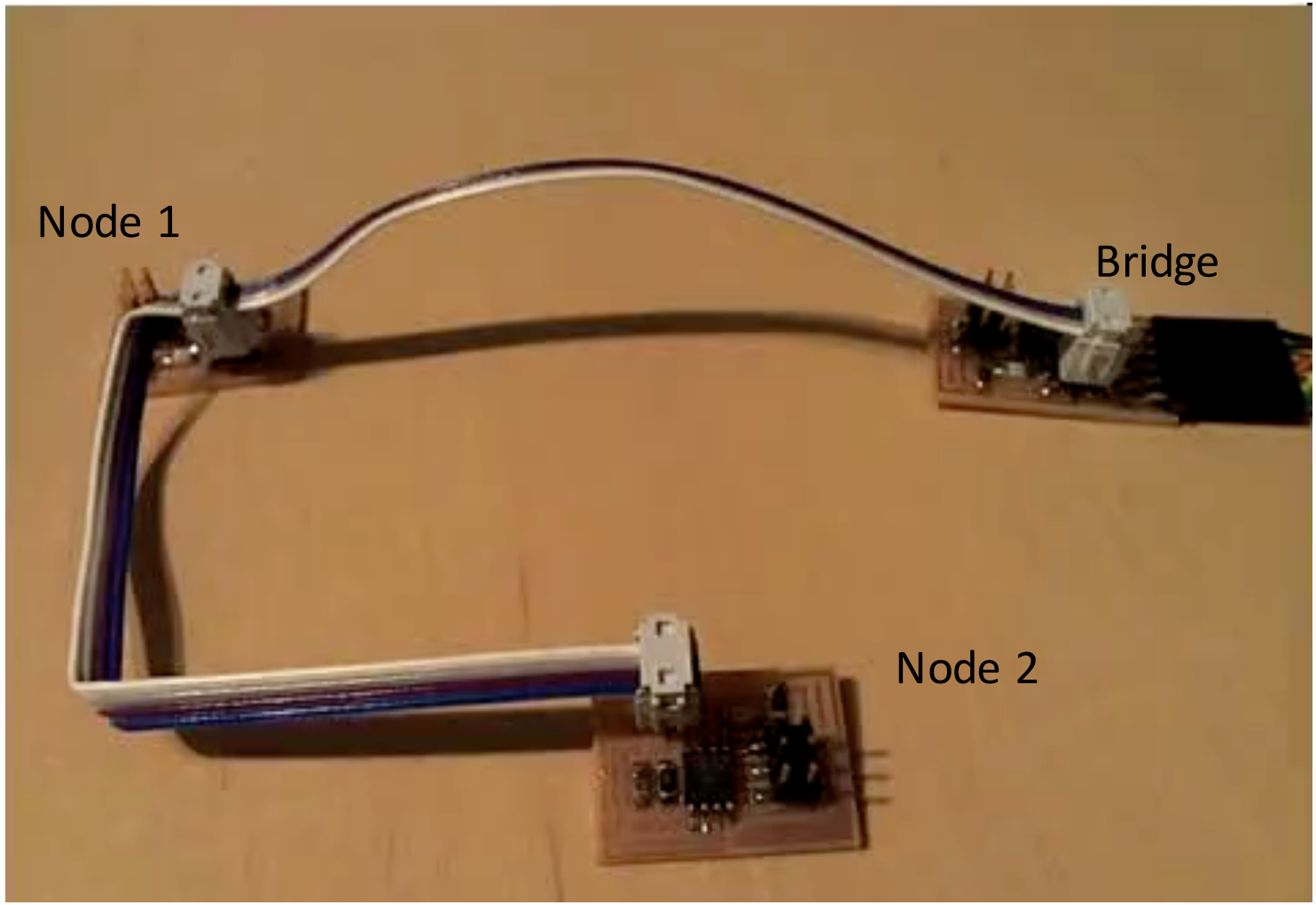
Software

If a microcontroller doesn't have a UART (or doesn't have enough), the serial interface can be **bit-banged - directly controlled by the processor.**

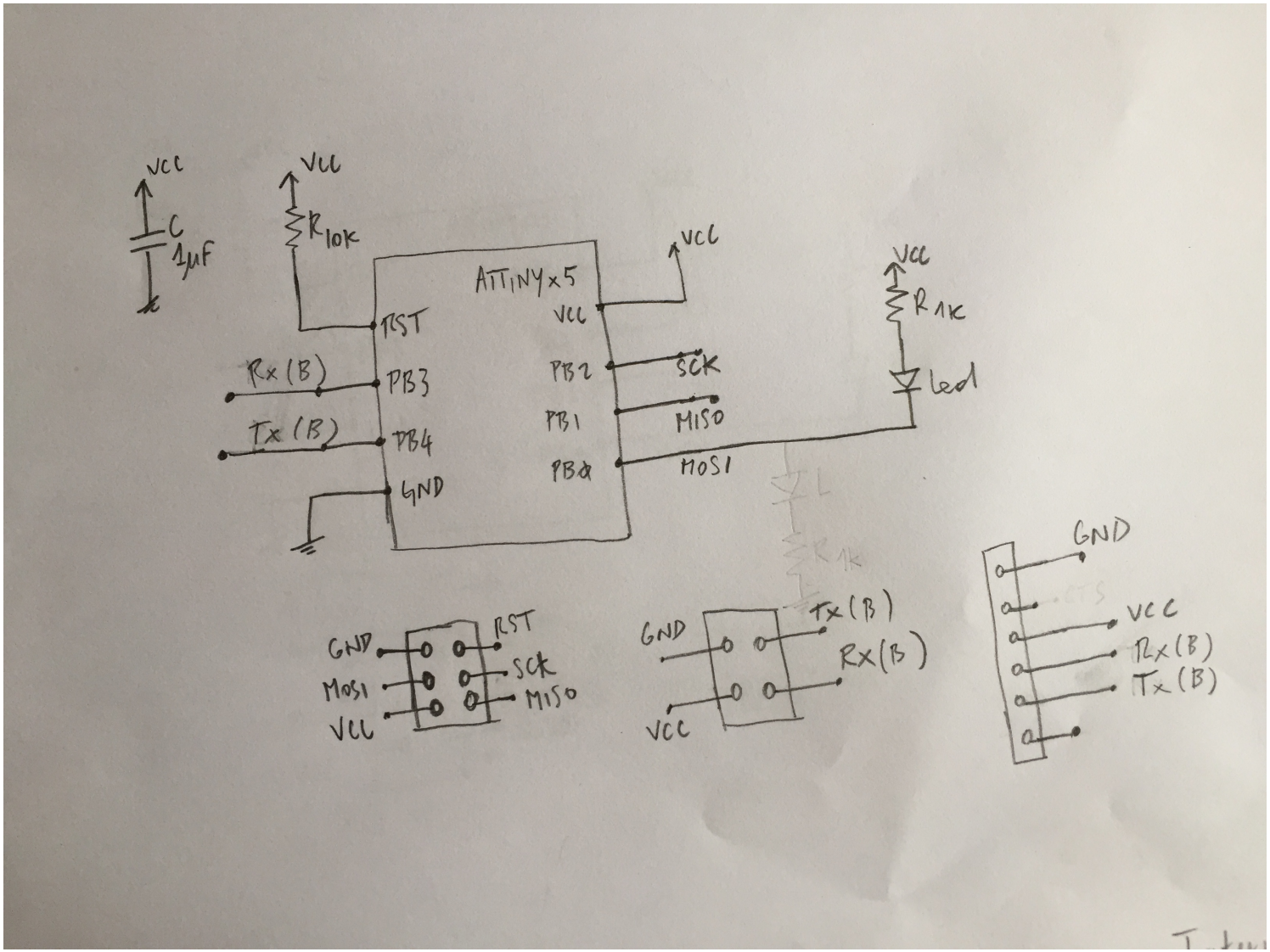
This is the approach of Arduino libraries like **SoftwareSerial library**.
It's the library that you use to read data from the Serial Monitor.

Reference: Niel's boards and codes
https://www.arduino.cc/en/Reference/SoftwareSerial
https://www.arduino.cc/en/Tutorial/SoftwareSerialExample

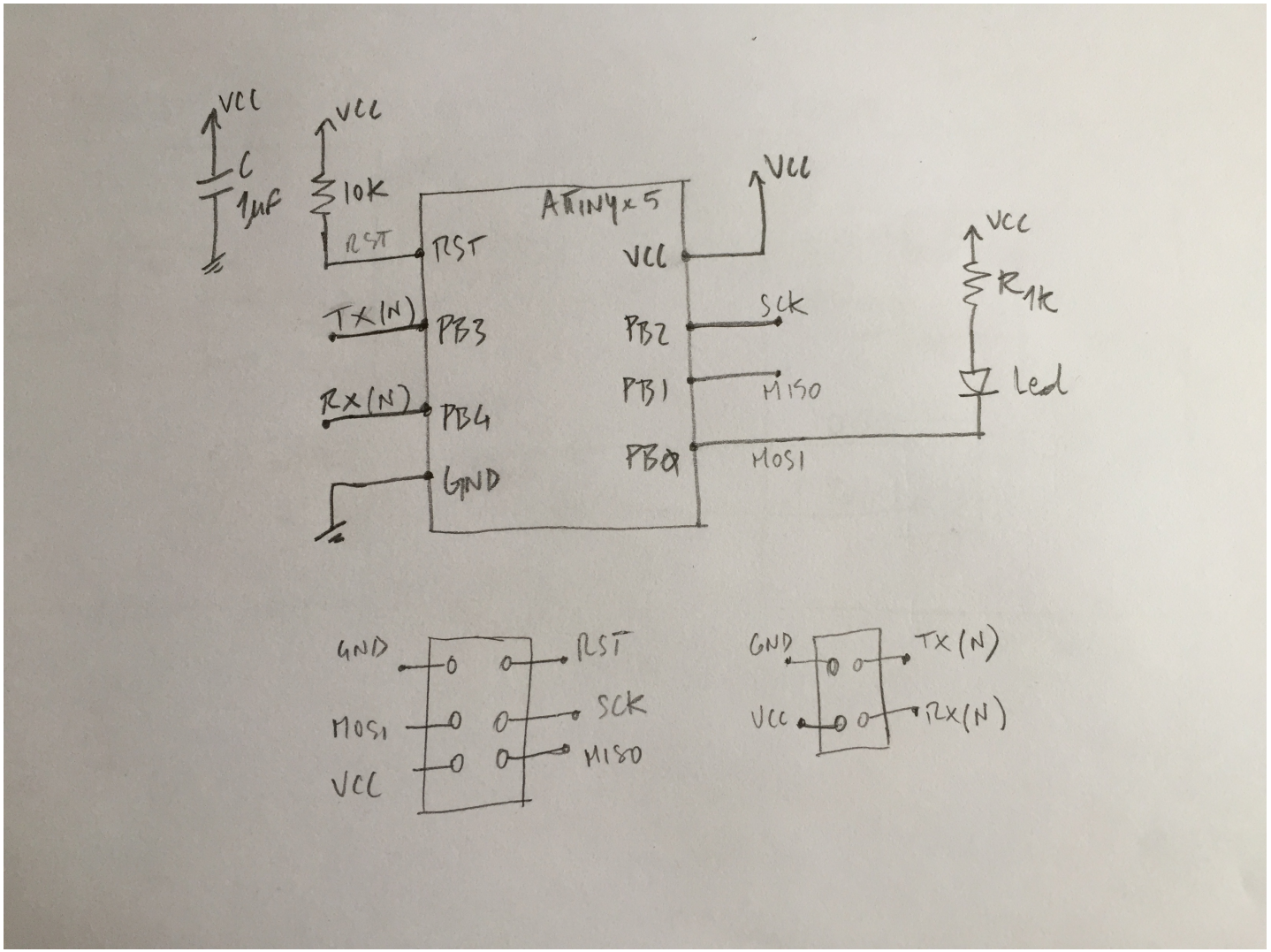
Asynchronous Serial Bus



Asynchronous Serial Bridge Schematic



Asynchronous Serial Node Schematic



Asynchronous Serial Bridge

network_serial_tx §

```
#include <SoftwareSerial.h>
#define rxPin 3
#define txPin 4

#define pin_led 0

SoftwareSerial serial(rxPin, txPin);

void setup() {
  pinMode(rxPin, INPUT);
  pinMode(txPin, OUTPUT);

  serial.begin(9600);
}

void loop(){

  serial.print(1);

  delay(2000);

  serial.print(2);

  delay(2000);

}
```

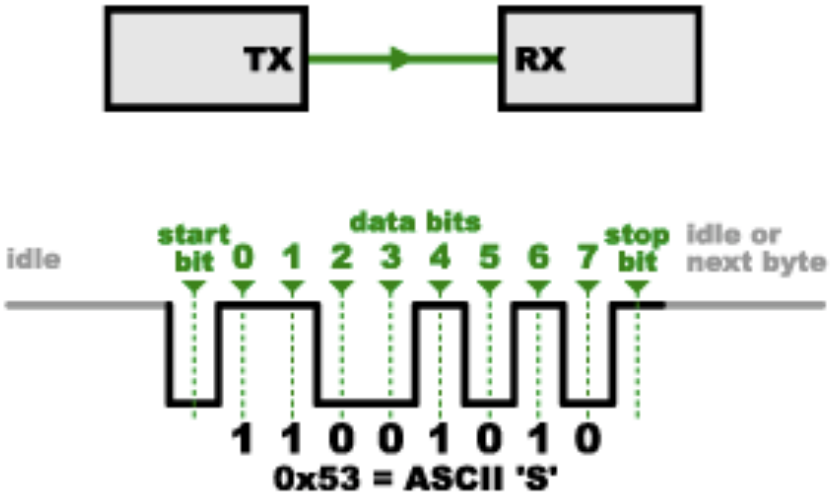
Asynchronous Serial Node 1

network_serial_n1 §

```
//NODE 1

#include <SoftwareSerial.h>
#define rxPin 4
#define txPin 3

#define pin_led 0

int v;

SoftwareSerial serial(rxPin, txPin);

void setup() {

  pinMode(rxPin, INPUT);
  pinMode(txPin, OUTPUT);

  pinMode(pin_led, OUTPUT);
  digitalWrite(pin_led, HIGH);

  serial.begin(9600);

}

void loop(){

  v = serial.read();

  if(v == '1'){
      digitalWrite(pin_led, HIGH);
  } else if(v == '2') {
      digitalWrite(pin_led,LOW);
  }

delay(1000);
}
```

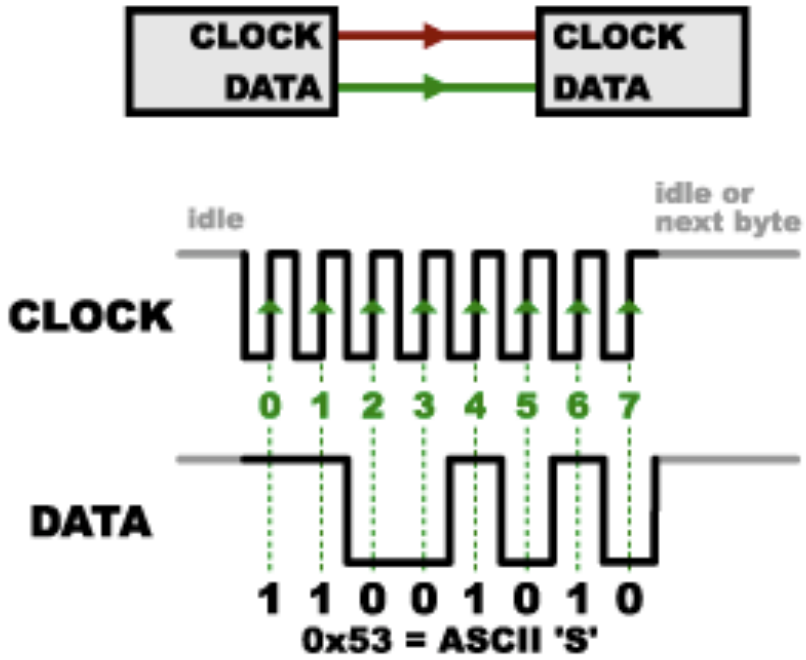
Why SPI (Serial Peripheral Iterface)?

What is wrong with Asynchronous?
Both sides must also agree on the transmission speed (such as 9600 bits per second) in advance.



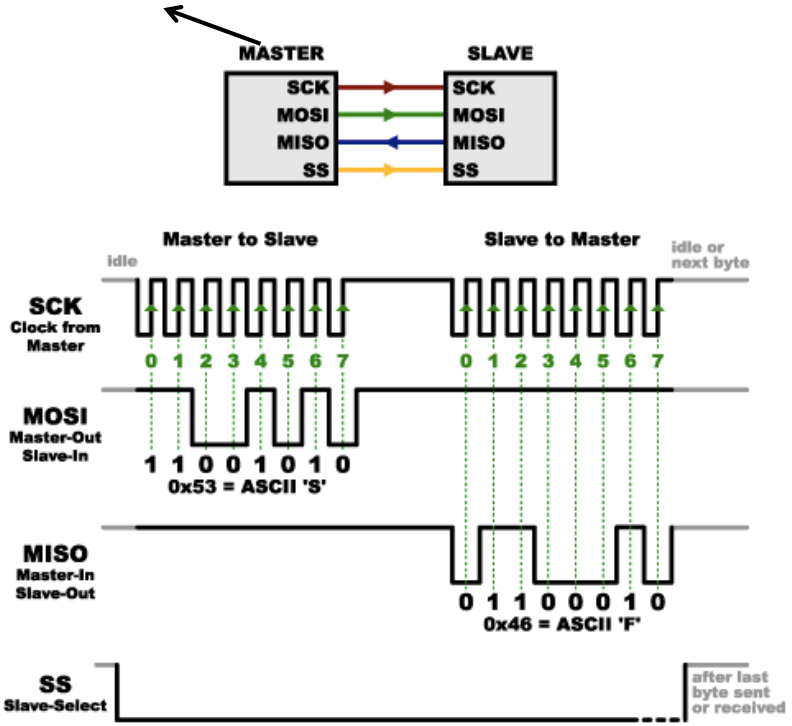
SPI – Serial Peripheral Interface

It's a "synchronous" data bus, which means that it uses separate lines for data and "clock" that keeps both sides in perfect sync.



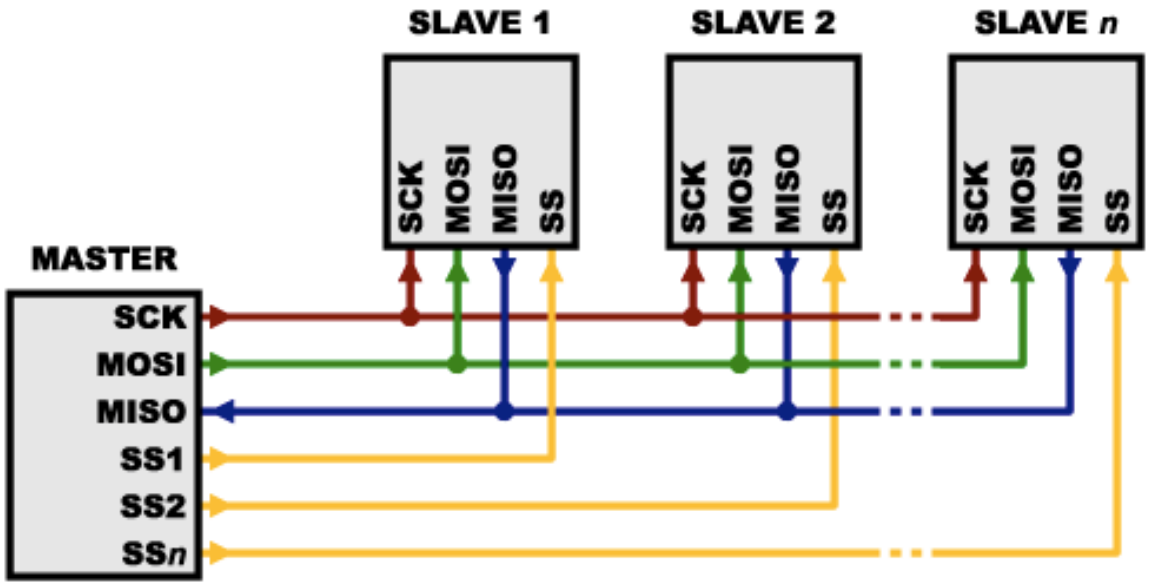
SPI – Serial Peripheral Interface

Master: it generates
the clock signal



SCK: Clock
MOSI: Master Out – Slave In
MISO: Master In – Slave Out
SS: to detect the Slave

SPI – multiple slaves



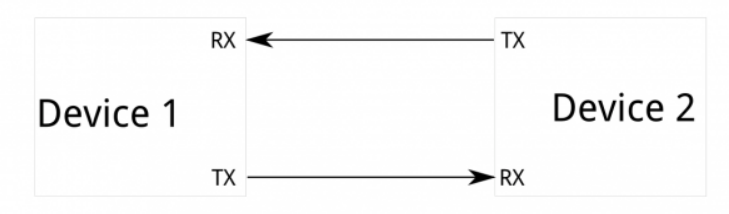
SPI

If you're using an Arduino, there are two ways you can communicate with SPI devices:
You can use the shiftIn() and shiftOut() commands.
Or you can use the SPI Library, but it's not developed for AttinyXX

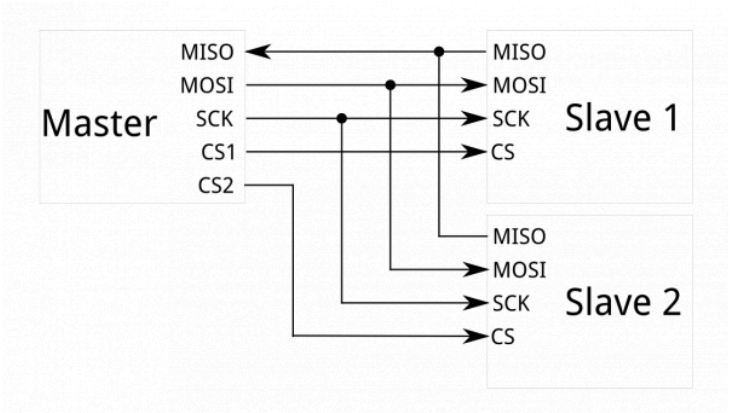
Example: http://www.gammon.com.au/spi

Why I2C?

What's Wrong with Serial Ports?

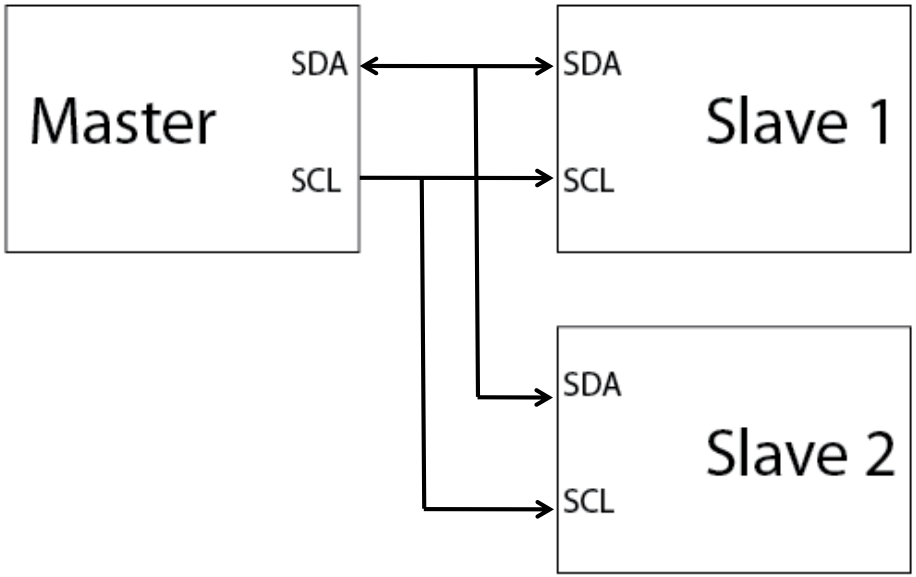


What's Wrong with SPI?



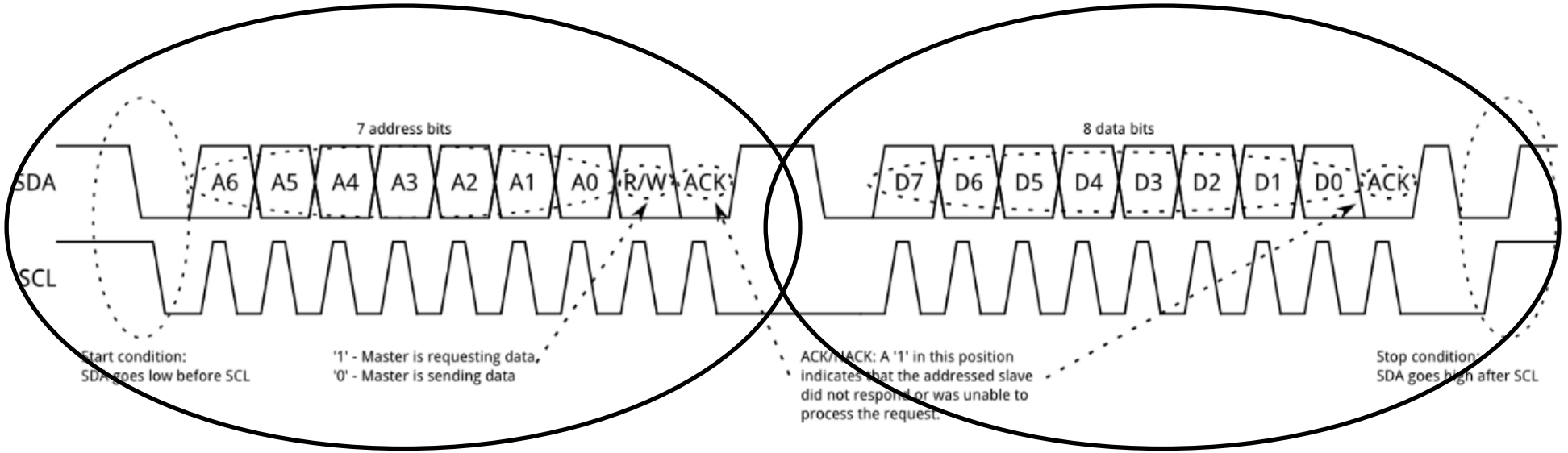
I2C

The best of the two worlds!



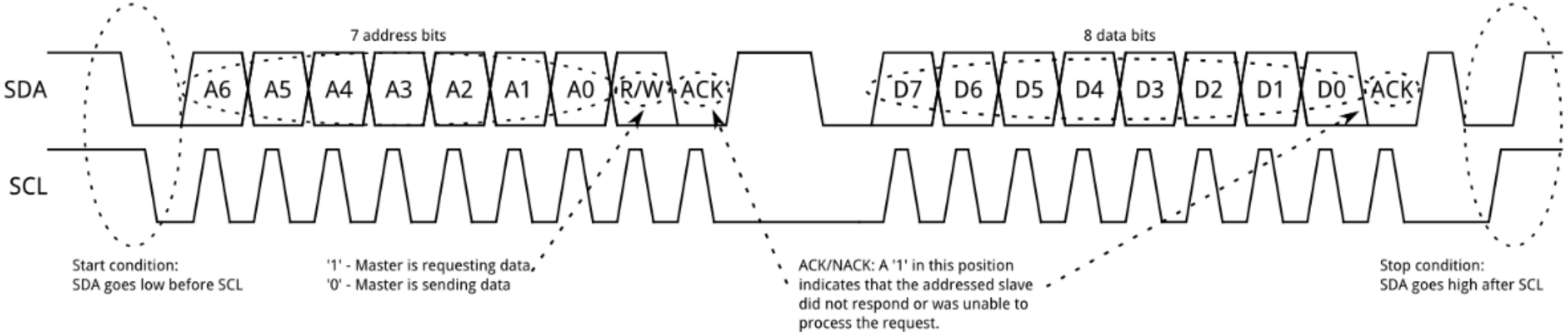
Those two wires can support up to 1008 slave devices.

I2C



Two types of frame:
- an address frame, where the master indicates the slave to which the message is being sent 2)
- one or more data frames, which are 8-bit data messages passed from master to slave or vice versa.

I2C



Start Condition:

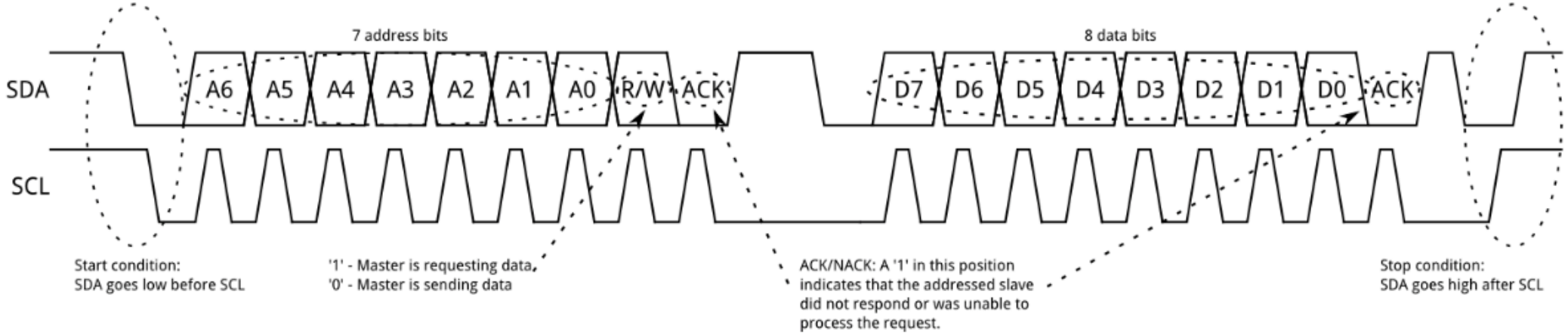
- the master device leaves SCL high and pulls SDA low
- all slave devices on notice that a transmission is about to start

Address Frame:

- always first in any new communication sequence

7-bit address, the address is clocked out most significant bit (MSB) first
followed by a R/W bit indicating whether this is a read (1) or write (0) operation.
NACK/ACK bit, for all frames (data or address).

I2C



Data Frames:
data can begin being transmitted. The master will simply continue generating clock pulses at a regular interval, and the data will be placed on SDA by either the master or the slave, depending on whether the R/W bit indicated a read or write operation.

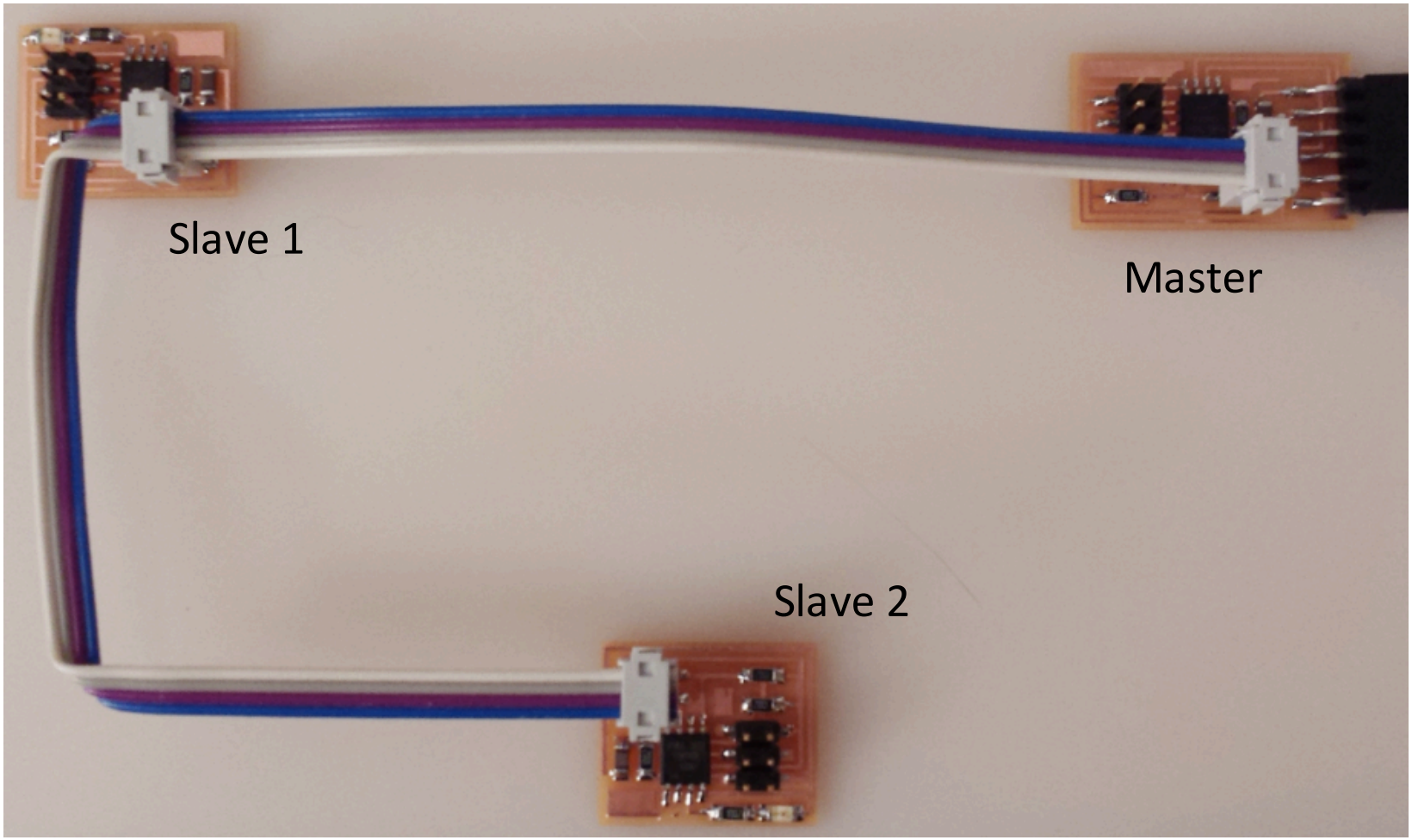
Stop condition:
Once all the data frames have been sent, the master will generate a stop condition. Stop conditions are defined by a 0->1 (low to high) transition on SDA *after a 0->1 transition on SCL, with SCL remaining high.*

I2C

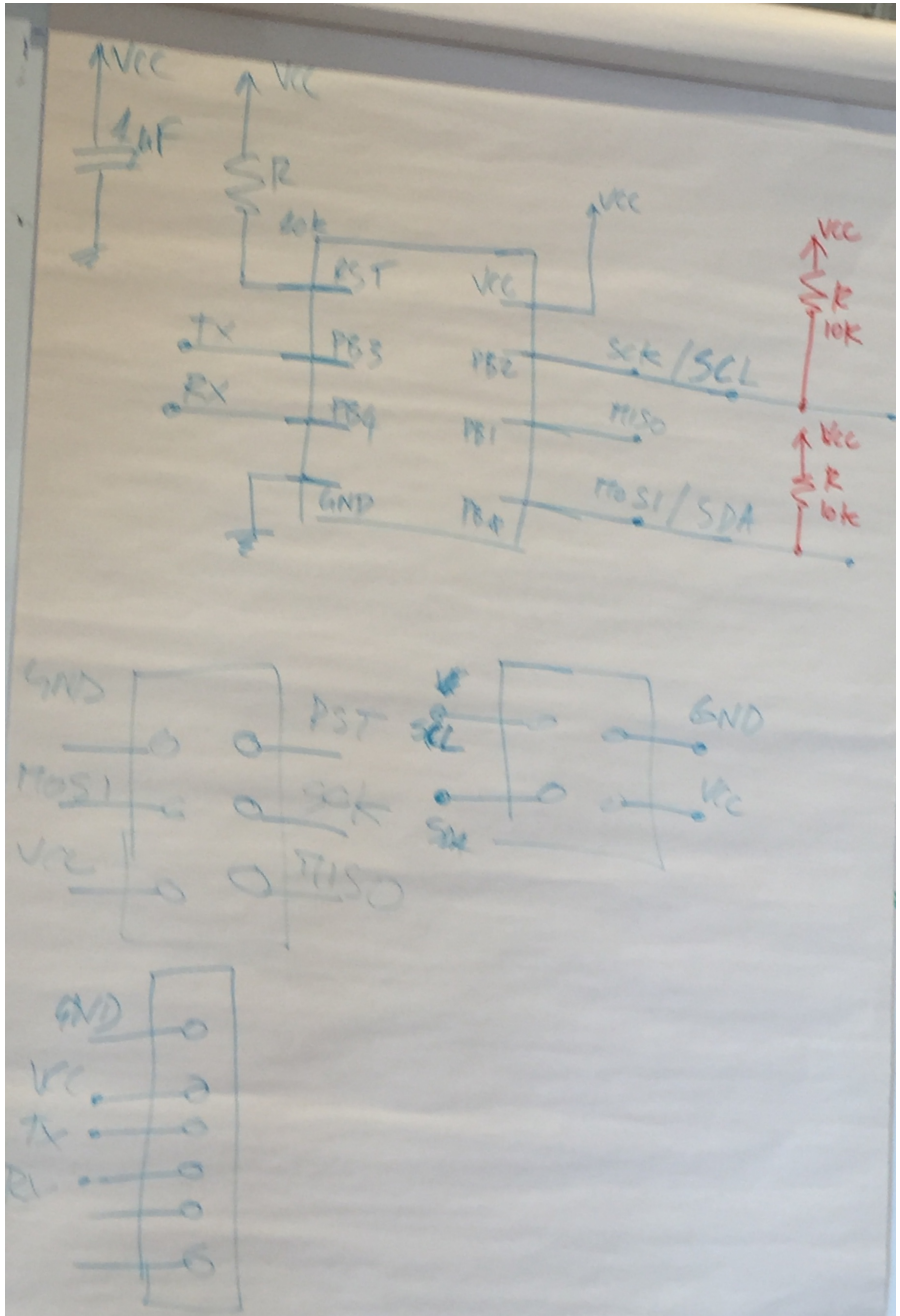
Reference:
Niel's boards and codes
Arduino Libraries, Attiny44:
TinyWireM
TinyWireS

Arduino Libraries, AtMega328P:
Wire
https://www.arduino.cc/en/Reference/Wire
https://www.arduino.cc/en/Tutorial/MasterReader
http://www.gammon.com.au/i2c

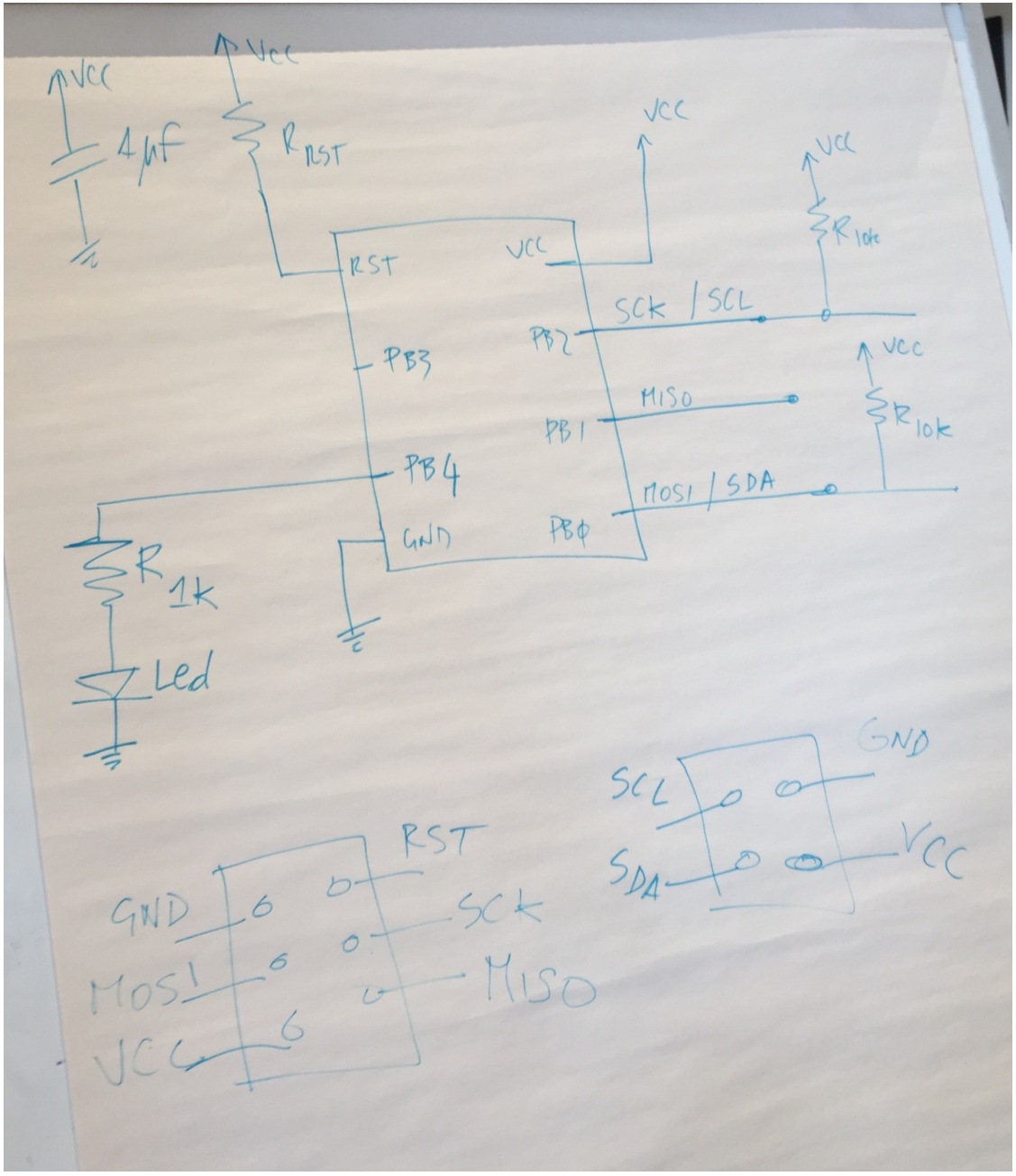
I2C - Network



I2C - Schematic Master



I2C - Schematic Slave



I2C - Master code

tiny45_M §

```
#include <TinyWireM.h>
#define device 0x01

void setup() {

TinyWireM.begin();

}

void loop() {

TinyWireM.beginTransmission(device);
TinyWireM.send(1);
TinyWireM.endTransmission();

delay(2000);

TinyWireM.beginTransmission(device);
TinyWireM.send(0);
TinyWireM.endTransmission();

delay(2000);

}
```

I2C - Slave code

tiny45_S

```
#include <TinyWireS.h>

#define output (4)
#define I2C_SLAVE_ADDR 0x01

volatile byte msg = 0;

void setup() {
  TinyWireS.begin(I2C_SLAVE_ADDR);
  pinMode(output, OUTPUT);
}


void loop() {

if (TinyWireS.available()){
  msg = TinyWireS.receive();
  }

if (msg == 1) {
    digitalWrite(output, HIGH);
    delay(2000);
     msg = 0;
     }
else if (msg == 0) {
    digitalWrite(output, LOW);
    }

delay(1000);
}
```