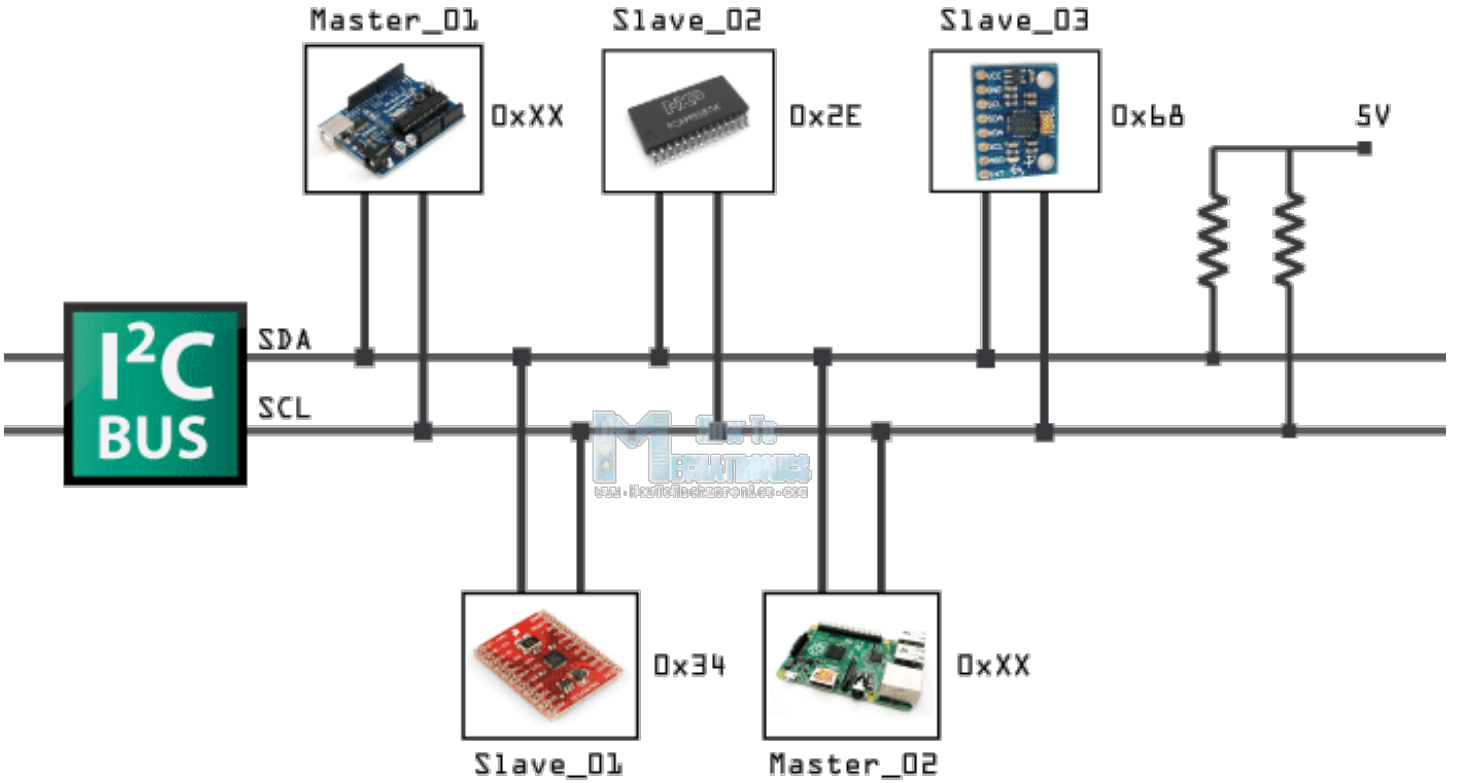
Awaiting :-)

Sensor/Actuator Communication by I2C

- Develop own I2C master slave nodes for
- your own realtime fieldbus
- develop communication scheme
- design & implement master and slaves
- (test timing)

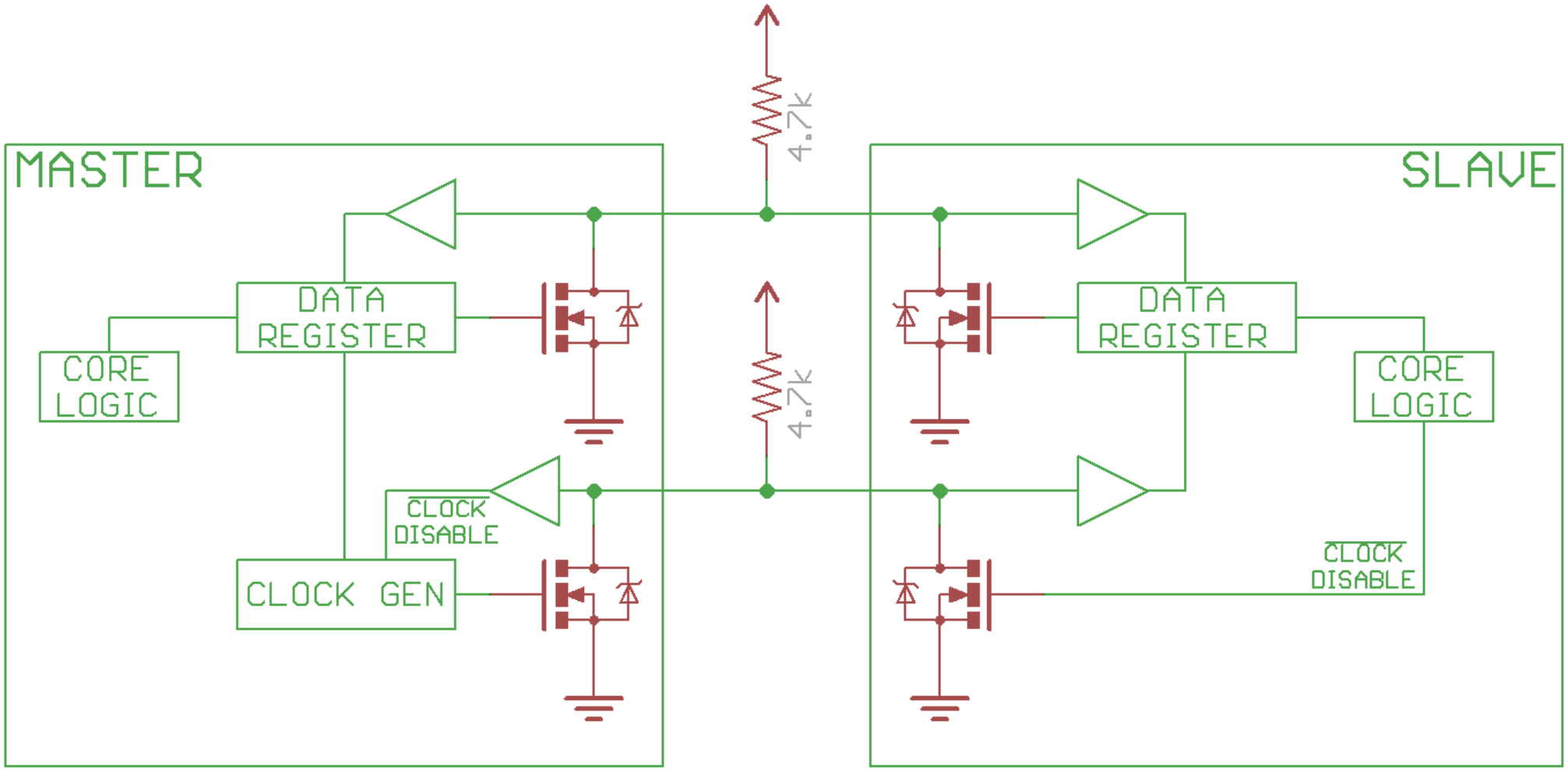


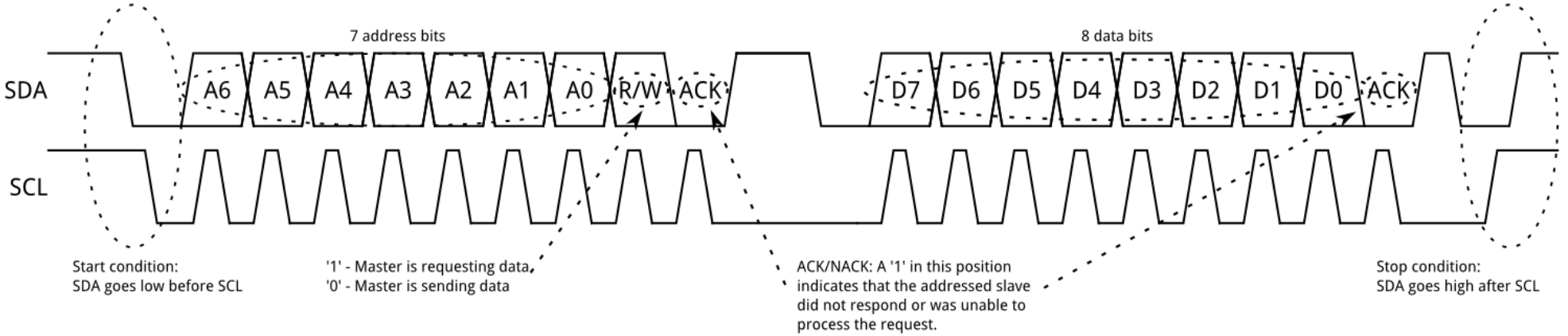
- http://i2c-bus.org/i2c-primer

I2C paradigm

- Master slave communication
  - only master can initiate communication
  - master can write data to slave
  - master can read data from slaves
- *Slave cant talk to slave !!!*
- *Slave cant initiate talk to master !!!*
- I2C can do multi master – but out of scope today)
- 100 kHz or 400 kHz

I2C electronics (high level is passive




7 address bits
8 data bits
SDA
SCL
A6
A5
A4
A3
A2
A1
A0
R/W
ACK
D7
D6
D5
D4
D3
D2
D1
D0
ACK
Start condition:
SDA goes low before SCL
'1' - Master is requesting data
'0' - Master is sending data
ACK/NACK: A '1' in this position
indicates that the addressed slave
did not respond or was unable to
process the request.
Stop condition:
SDA goes high after SCL

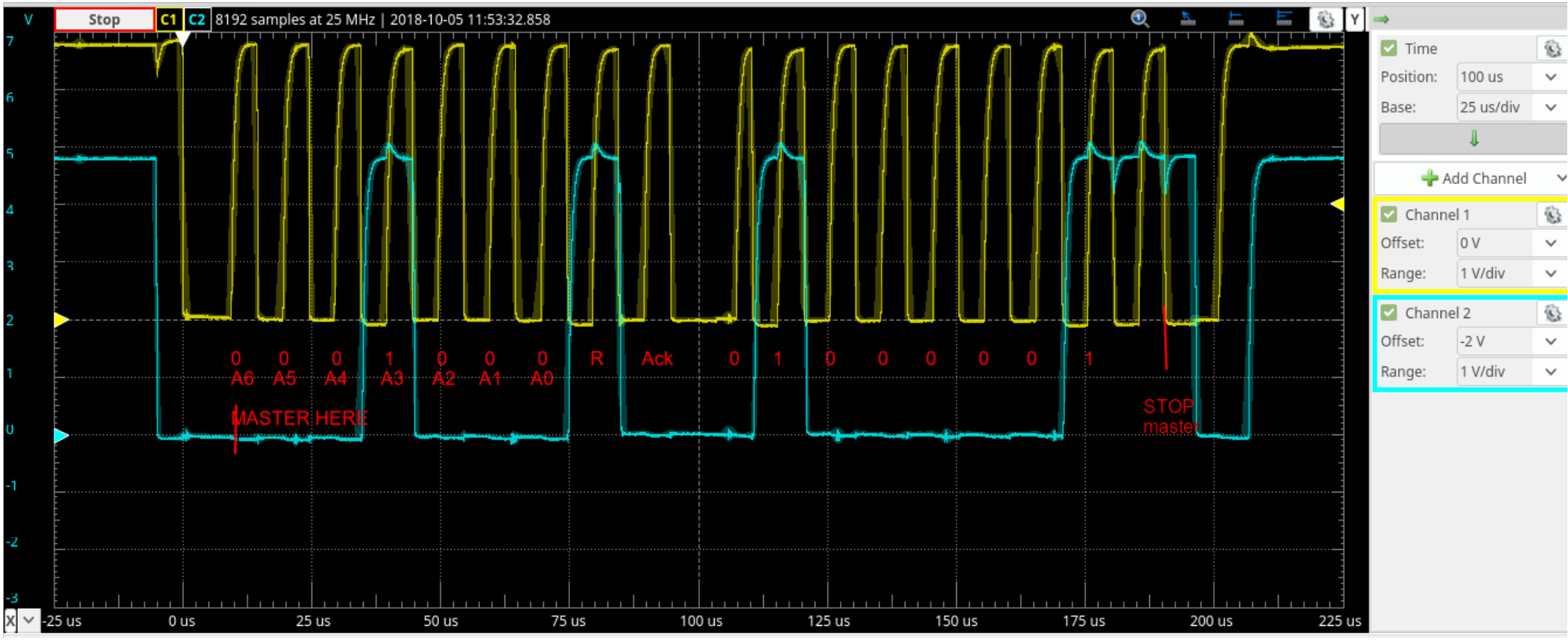
the Hidden Protocol

- Master and slave need to have negotiated a protocol
- Always:
  - master tells either
  - I will write/Send data to you
  - or
  - I want to receive data from you
- In either case YOU have to define a protocol based on
  - Master always set agenda
  - Slave has to obey
  - Can be simple or complex – it’s up to you

```
// Wire Slave Sender
#include <Wire.h>

void setup() {
  Wire.begin(8);                // join i2c
bus with address #8
  Wire.onRequest(requestEvent); // register
event
}

void loop() {
  delay(100);
}

// function that executes whenever data is
requested
// this function is registered as an event

void requestEvent() {
  // B
  Wire.write(0x42);
  // as expected by master
}
```

```
 // Wire Master Reader
#include <Wire.h>

void setup() {
  Wire.begin();
  // join i2c bus (address optional for
master)
  Serial.begin(9600);  // start serial for
output
}

void loop() {
// request 1 bytes from slave device #8
  Wire.requestFrom(8, 1);
// slave may send less than requested
  while (Wire.available()) {
    char c = Wire.read();
    Serial.print(c);         // print the
character
  }

  delay(500);
}
```

Master req 1 byte from slave #8 (get a A 0x14)

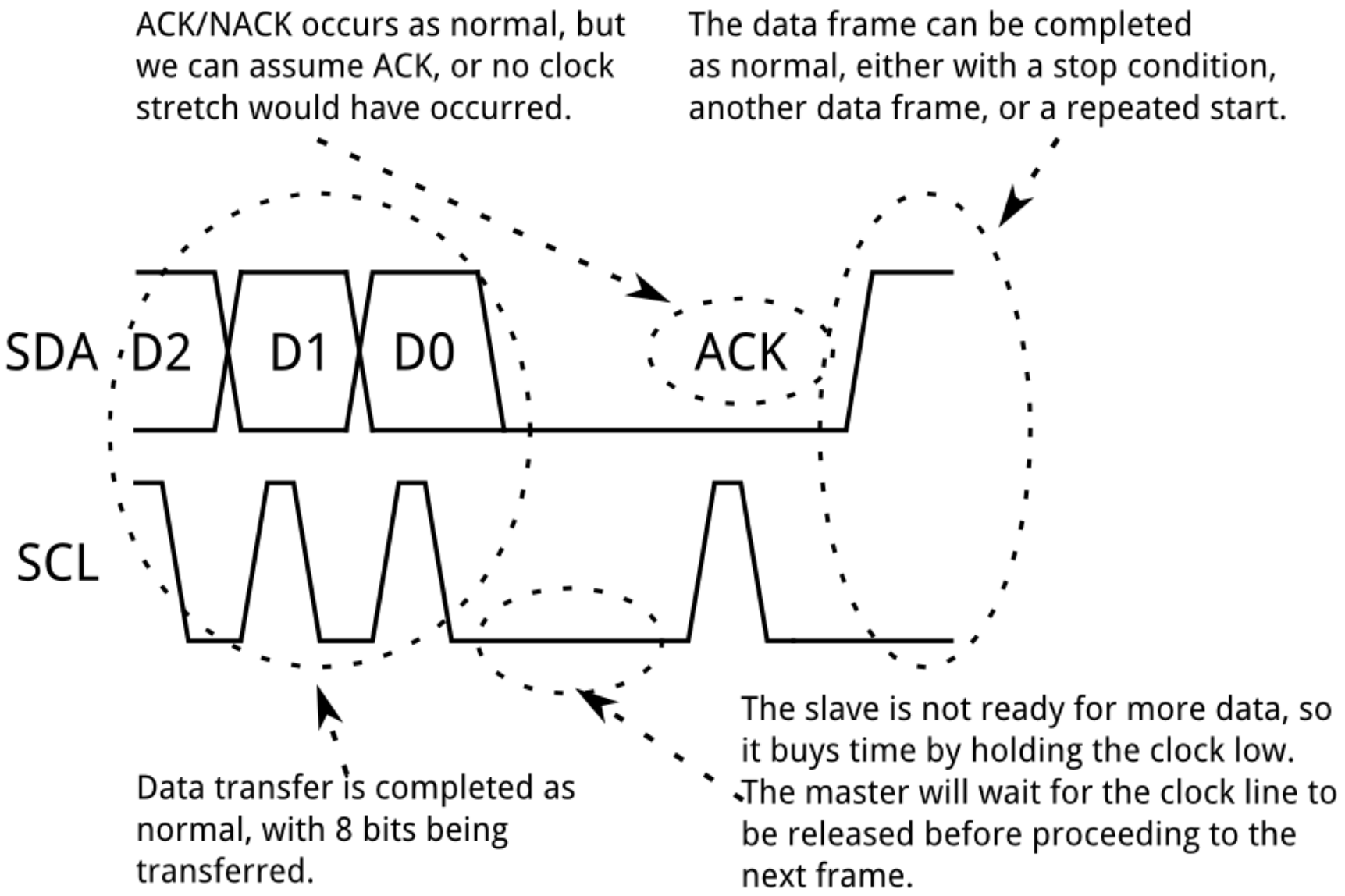


simple master read protocol

```
#include <Wire.h>

void setup() {
  Wire.begin();
Serial.begin(9600);
}
void loop() {
  Wire.requestFrom(8, 6);
// request 6 bytes from slave #8

  while (Wire.available()) {
  // slave may send less than requested

    char c = Wire.read();
    Serial.print(c);
  }

  delay(500);
}
```

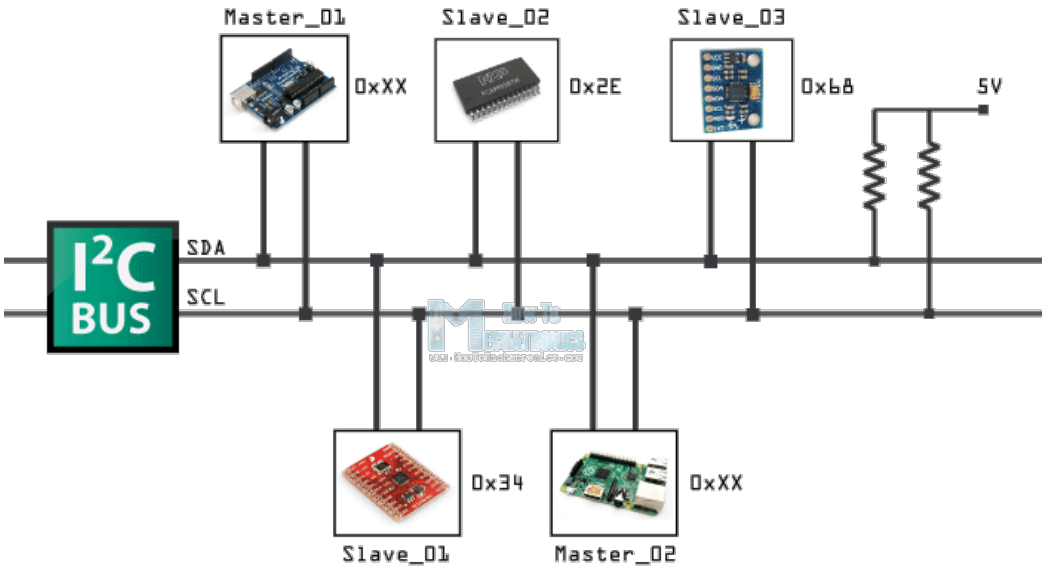
```
#include <Wire.h>

void setup() {
  Wire.begin(8); // slave addr  #8
  Wire.onRequest(requestEvent);
}

void loop() {
  delay(100);
}

// function that executes whenever
// data is requested by master

void requestEvent() {
   Wire.write("123456");
   // respond with message of 6 bytes
   // as expected by master
}
```

Clock stretching – *slave buys time*



Spec of today mini project

- Two or more Arduinos
  - connect SDA and SCL lines
  - if different power the also connect GND
  - Pullup is integrated Arduino but you might add two 10 kOhms just to be sure



a comment

- You cant specify a read address in a read call from master.
- you can only specify id of slave.

- so it’s real simple :-)

Slave memory map

- adr 0-3 : “read” analog read a0-a3
- adr 4-7 : “read” digital read pin 8,9,10,11
  - initalised as INPUT_PULLUP
- adr 4-7 : “write” digital write pin 4,5,6,7
  - initalised as OUTPUT
- adr 8 : “write” Serial.print()

- adr 4-7 is used as write and read memory :-)
  - just to save memory map

typical read from slave

- master write ID of register to a dedicated register reference register
  - write(<i2c id><reg ref register>< value>)
- and issues a read just after
  - read(<i2c id> <&dest>)

- reg ref register is at adress 39
- we want to read register no 3

```
char dst;
Wire.”write”<i2c ID>, 39, 3);
Wire.”read”(<i2c UD>, &dst);
```

Spec II

- analog ports at slave - read only
- digital port at slave - read and write
- serial port at slave - write only (Serial.print)

write to slave in C

```
void xxx(char regNo, char *p)  {
int i;
 Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.write(regNo);
  for(int i=0; i<length; i++) {
    Wire.write(*p);
    p++;
  }
  Wire.endTransmission();
}
```

slave call back

```
#include <Wire.h>

void setup() {
  Wire.begin(8);                // join i2c bus with address #8
  Wire.onRequest(requestEvent); // register event
}

void loop() {
  delay(100);
}

// function that executes whenever data is requested by master
// this function is registered as an event, see setup()
void requestEvent() {
  Wire.write("hello "); // respond with message of 6 bytes
  // as expected by master
}
```

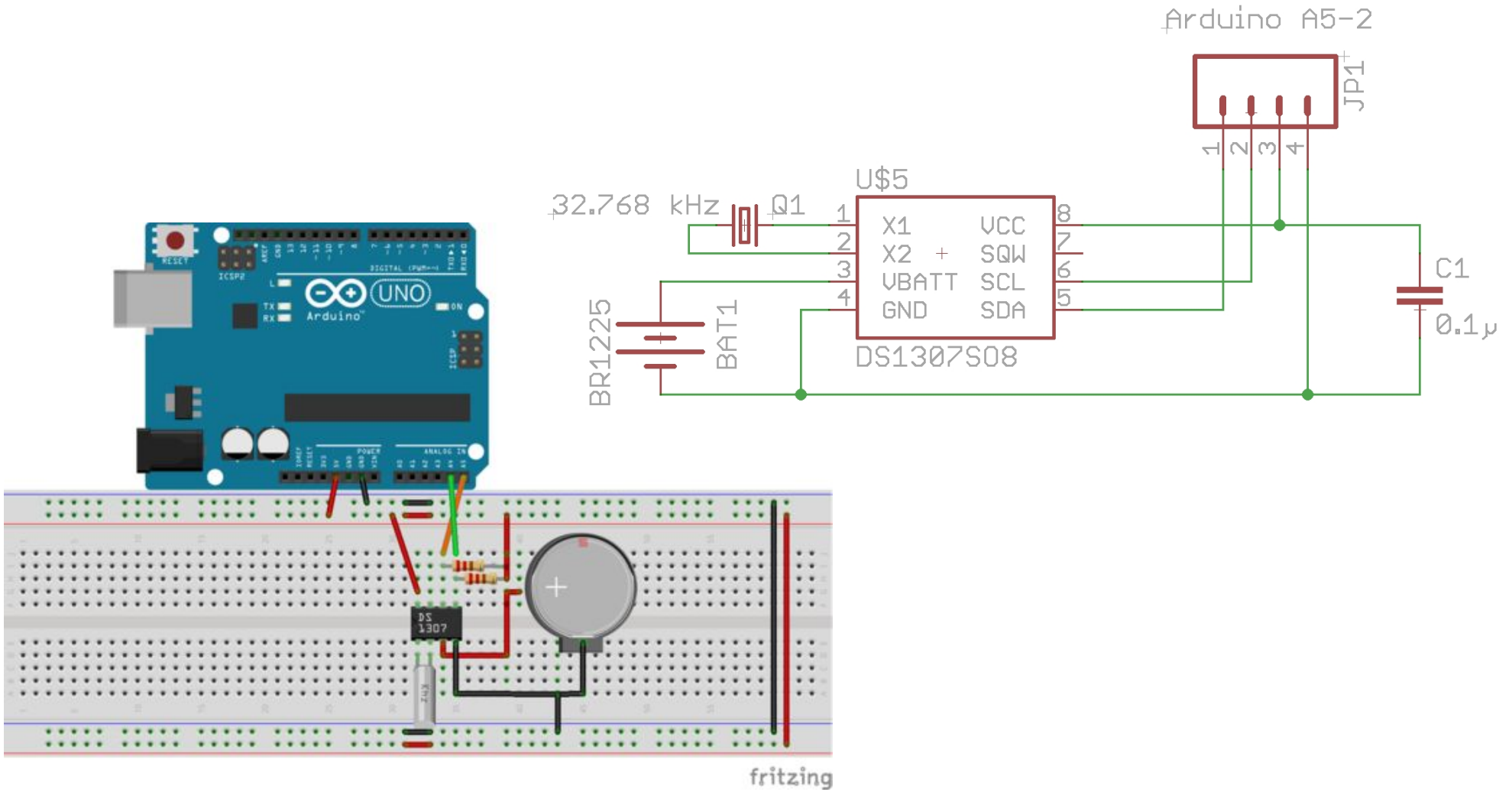
exercise II – DS 1307 real time clock

- Pick up a DS 1307 and 32 kHz XTAL at lab
- Do the mockup
- Get it up and running

NB at instructables below be aware of how they use

http://www.instructables.com/id/Arduino-Real-Time-Clock-DS1307/

DS 1307


Arduino A5-2
JP1
1
2
3
4
U$5
32.768 kHz
Q1
X1
X2
VBATT
GND
VCC
SQW
SCL
SDA
DS1307SO8
BR1225
BAT1
C1
0.1µ
UNO
Arduino
DS
1307
fritzing

DS 1307 real time clock(slave addr 0x62)

Simple R/W on all addresses

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/ AM							
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

```
char readASingleByte(char regNo)
{
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.write(regNo);
  Wire.endTransmission();
  Wire.requestFrom(DS1307_I2C_ADDRESS, 1);
  return Wire.read();
}


// you can read more than one – its up to you
```