Week 13: Networking and Communications

29.4. 2015

This week we will learn about networking and communications.

The agenda:
http://academy.cba.mit.edu/classes/networking_communications/index.html

```
purposes
   location
   parallelism
   modularity
   interference

serial
   asynchronous
      RS-232: http://www.maximintegrated.com/en/app-
notes/index.mvp/id/723
RS-422:  http://www.maximintegrated.com/en/app-notes/index.mvp/id/723
RS-485: http://www.maximintegrated.com/en/app-notes/index.mvp/id/723

      components video
      hello.bus.45.bridge.cad board traces interior
      hello.bus.45.node.cad board traces interior
      hello.bus.45.c makefile
   I2C, TWI:  http://www.nxp.com/documents/user_manual/UM10204.pdf
      TWI master slave
      USI master slave
      software master slave
      library
      hello.I2C.45.bridge.cad board traces interior
      hello.I2C.45.node.cad board traces interior
      components programming
   SPI: http://www.atmel.com/images/doc2585.pdf
      library
   USB: http://www.usb.org/home
      Hardware: http://www.digikey.com/product-detail/en/ATMEGA16U2-
AU/ATMEGA16U2-AU-ND
         AVR LUFA
         STM32
      Software:
http://academy.cba.mit.edu/classes/embedded_programming/hello.ISP.44.
png
         V-USB: https://www.obdev.at/products/vusb/index.html

OSI layers:  http://www.iso.org/iso/home.htm
   7: application (HTTP)
   6: presentation (SSL)
   5: session (RPC)
   4: transport (TCP, UDP)
   3: network (IP)
   2: data link (MAC)
   1: physical (PHY)

physical media:
http://www.cambridge.org/us/academic/subjects/physics/general-and-
classical-physics/physics-information-technology
   capacity
```

```
      bandwidth * log_2 (signal/noise)
   wired
      single-ended, differential, powerline
      open collector, open drain
      transmission (pass) gate, tri-state
      transmission line
      waveguide
      TIA RS232, 422, 485
      802.3 ethernet
         chip module
      SONET optical fiber
   wireless
      RF
         FCC Part 15 ISM
         802.11 Wi-Fi
         802.15 ZigBee
         6LoWPAN
         Bluetooth
      optical
         transmitter receiver
      acoustic

modulation: http://www.crcpress.com/product/isbn/9780849309670
   PCM: Pulse-Code Modulation
   PPM: Pulse-Position Modulation
   OOK: On-Off Keying
   FSK: Frequency-Shift Keying
   BPSK: Binary Phase-Shift Keying
   QAM: Quadrature Amplitude Modulation
   OFDM: Orthogonal Frequency-Division Multiplexing
   FHSS: Frequency-Hopping Spread Spectrum
   DSSS: Direct-Sequence Spread Spectrum
   UWB:  Ultra-WideBand

channel sharing: http://authors.phptr.com/tanenbaumcn4/
   ALOHA
   Master-Slave
   Token Ring
   TDMA: Time-Division Multiple Access
   FDMA: Frequency-Divsion Multiple Access
   CSMA: Carrier-Sense Multiple Access -
      CD: Collision Detection -
      CA: Collision Avoidance -
      1-persistent: transmit when clear -
      nonpersistent: random backoff -
      p-persistent: probability to transmit -
   CDMA: Code-Division Multiple Access
   MIMO: Multiple-Input Multiple-Output
   "PDMA": Physical-Division Multiple Access

errors:
http://www.cambridge.org/us/academic/subjects/physics/general-and-
classical-physics/physics-information-technology
   detection, correction
   block, convolution codes
   parity, checksum, Hamming, Reed-Solomon, Turbo

networking: http://authors.phptr.com/tanenbaumcn4/
   Internet protocols: http://www.ietf.org/
        IPv4:  http://www.ietf.org/rfc/rfc0791.txt
        IPv6: http://www.ietf.org/rfc/rfc2460.txt
        DNS: http://www.ietf.org/rfc/rfc1035.txt
        DHCP NAT private
```

UDP, TCP
HTTP: http://www.ietf.org/rfc/rfc2616.txt
BGP: http://www.ietf.org/rfc/rfc4271.txt
AODV ROLL
sockets
udpsnd.py udprcv.py
udpsnd.c udprcv.c
Wireshark: https://www.wireshark.org/ - Sniffer
SLIP: http://www.ietf.org/rfc/rfc1055.txt
Slattach:
http://manpages.ubuntu.com/manpages/trusty/en/man8/slattach.8.html
route:
http://manpages.ubuntu.com/manpages/trusty/man8/route.8.html

hello.bus.45.SLIP.c makefile udp_slip.py video:
http://academy.cba.mit.edu/classes/networking_communications/SLIP/hello.bus.45.SLIP.mp4
Internet 0: http://cba.mit.edu/docs/papers/06.09.i0.pdf
asynchronous packet automata (APA)
source routing + network coordinates + back-pressure flow-control + synchronous communication
components video
apa.ftdi.cad board traces interior apa.ftdi.c makefile
apa.io.cad board traces interior apa.io.c makefile
apa.c apa.h
apa.py

RF
Radios: http://www.arrl.org/shop/What-s-New
oscillator, mixer, PA, LNA, IF, I/Q, demod, baseband, filters
antennas: http://eu.wiley.com/WileyCDA/WileyTitle/productCd-047166782X.html
Q, antenna gain, impedance matching
FabFi: https://code.google.com/p/fabfi/wiki/FabFi
single-chip
MICRF (300-470 MHz)
Transmitter: http://www.digikey.com/product-detail/en/MICRF102YM/576-1338-ND
Receiver: http://www.digikey.com/product-detail/en/MICRF008YM/576-1961-5-ND

Arecibo, Puerto Rico:
https://www.google.com/search?q=arecibo+puerto+rico&tbm=isch&imgil=a2QvJ4usEQvf5M%253A%253BicEBUZRVdHppvM%253Bhttp%25253A%25252F%25252Fwww.topuertorico.org%25252Fcity%25252Farecibo.shtml&source=iu&pf=m&fir=a2QvJ4usEQvf5M%253A%252CicEBUZRVdHppvM%252C_&usg=__3TAkN7DBgnYe7ryYWRgznGJQKUw%3D&biw=1334&bih=862&ved=0CDwQyjc&ei=X_pAVYDNIYTUasaSgLgF#imgrc=a2QvJ4usEQvf5M%253A%3BicEBUZRVdHppvM%3Bhttp%253A%252F%252Fwww.topuertorico.org%252Fimg%252Fhp-tescop.jpg%3Bhttp%253A%252F%252Fwww.topuertorico.org%252Fcity%252Farecibo.shtml%3B400%3B300

http://en.wikipedia.org/wiki/Arecibo,_Puerto_Rico

MRF49XA (433/868/915 MHz):
http://www.microchip.com/wwwproducts/Devices.aspx?product=MRF49XA
chip board module

nRF905 (433/868/915 MHz): http://www.digikey.com/product-detail/en/NRF905/1490-1028-ND - https://github.com/zkemble/nRF905

chip module library
nRF24L01+ (2.4 GHz ISM): http://www.digikey.com/product-detail/en/NRF905/1490-1028-ND

chip module library
ESP8266 (2.4 GHz Wi-Fi)
chip module commands library
HC-05 (2.4 GHz Bluetooth): http://www.amazon.com/JBtek-Wireless-Bluetooth-Transceiver-Arduino/dp/B00L083QAC
chip module library : https://github.com/jdunmire/HC05
software radio: http://gnuradio.org/redmine/projects/gnuradio/wiki

Very active user community around this chip.

Assignment

To design and build a wired &/or wireless network connecting at least two processors.

Use light and sound to communicate.

Discussion on project and how to use the networking assignment to develop the final project:

3 outputs – 3 flowers

Write a use case
Ignore the sensor on the output board – and make the input board a master. Create another output board (or 2) and connect output boards with a cable and multiple connectors.

Multiple inputs (3) – mixing 3 tones, melody, changing rythm or speed

How are they going to communicate and what are the messages going to be.
Write a script for the communication.

Programming lesson:

Reset = 0

J2 FTDI =
IC1 t44 = Microcontroller
C1 1uf=Capacitor
R1 10k=Resistor
XTAL1 20 MHz = Crystal = time, is there because of the microcontroller and the tasks that it has to perform

This was the first board we made.

Arduino environment

Baud = how many bits per second

9600 command rate with Neil

Comment in the code

Open code in “Text edit”

Parameters tell function what to do – we need to give it the value

All the pins have got 3 registers:
Get char function is a general function
With the parameters we tell which pins we want to talk to
Where to look and where to put it
Serial_pin
Serial_pin_in
Serial_pin_out

When using Neil's code you have to use "make" etc
Using Arduio is simpler and it has a good reference

Get char – receives a character
Put char – sends one character

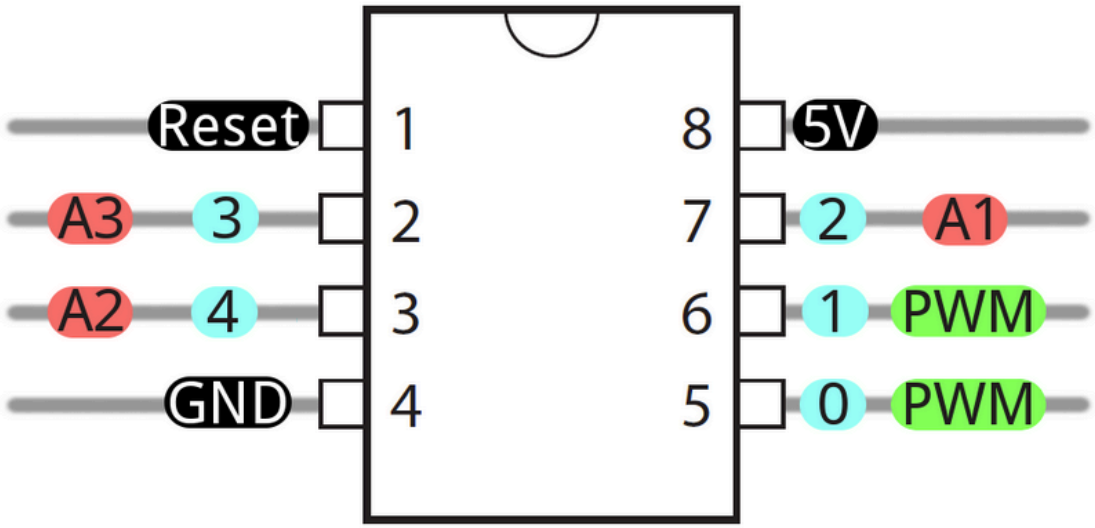
Put string – sends many characters

Variable – is like a container, with a name. Stores information

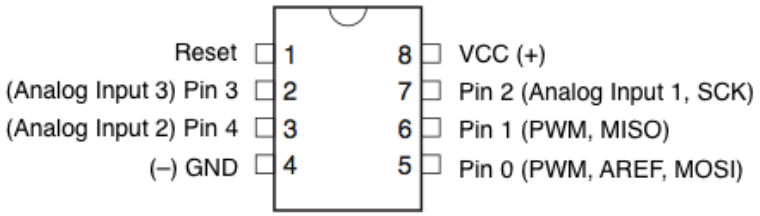
Declaration of the variable you need are written before the setup (in Arduino)
Labelling of the containers is done before setup

Programming in Arduino: http://highlowtech.org/?p=1695

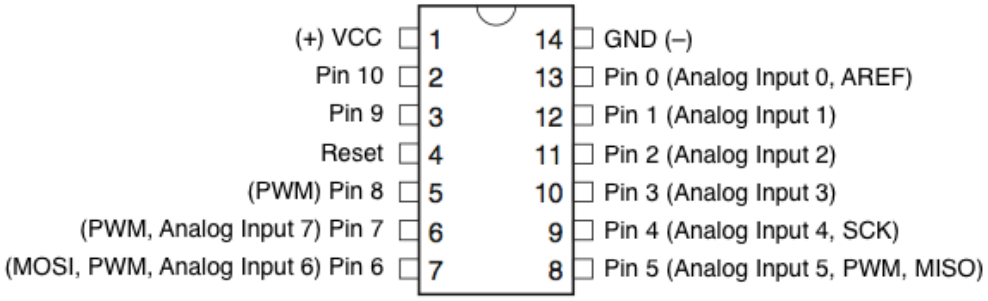
Tiny AVR Programmer Hookup Guide:
https://learn.sparkfun.com/tutorials/tiny-avr-programmer-hookup-guide



ATtiny45 / ATtiny85

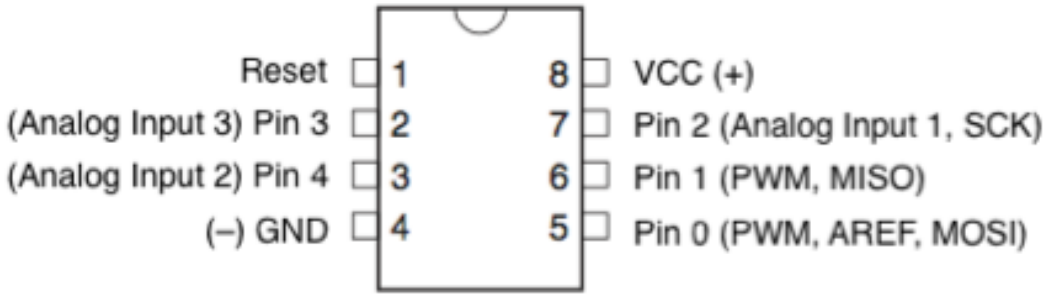


ATtiny44 / ATtiny84



Microcontroller 1 (input board):

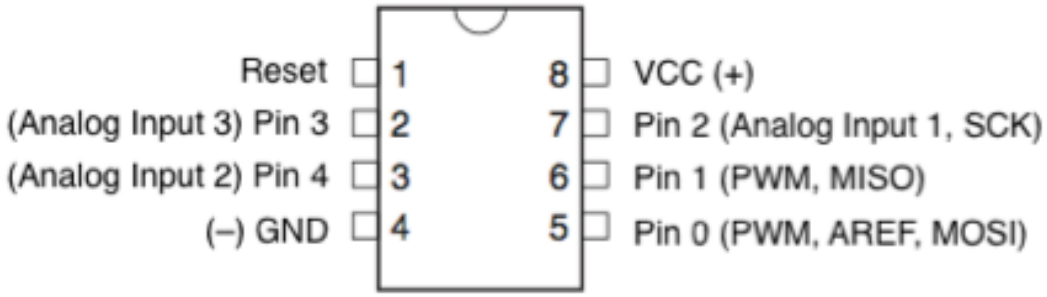
Attiny 45



Connector

Microcontroller 2 – (output board):

Attiny 45



Connector

```
int addup (int a, int b) [return a+b]

index=0

While = loop

See Learning – Reference in Arduino to clarify terms and functions

Parenthesis – ()
Curly braces – {}

While (1) – reads as true, keeps running

Buffer = array
Buffer[index] = char
Index = index + 1
```

Using Neil's code in Arduino:

```
Serial is a library that is included by default in Arduino
This is using the internal library

Standard library = SoftwareSerial

Reference – Libraries, for information

Start with the library and the library gives you a set of functions
It needs a setup

SoftwareSerial mySerial (10, 11) (Those are arduino serial, we need
to match this with
SoftwareSerial mySerial (1, 0)

Tx – receive
Rx – transmit
```

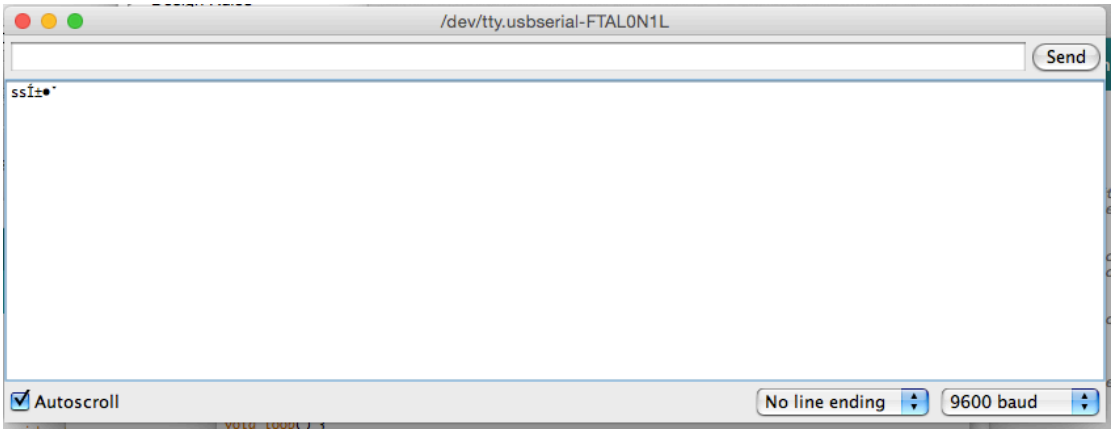
Test: sketch_serial_Test

```
Sketch – and set Board

Burn bootloader
```

http://www.arduino.cc/en/Reference/Libraries

```
Copy from Example:

#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX
into parameter section


Copy:  mySerial.begin(4800); into void setup

Change figure to 9600 (speed of talking)

Copy: if (mySerial.available())
    mySerial.write(mySerial.read());
and paste into void loop
```

Start Serial Monitor 
Set 9600 baud



Type a letter in send line – one letter will pop up in the window below, but if a string of letters is typed only the first letter will appear

Now the following code is entered:

```
  if (mySerial.available()) {
  char chr = mySerial.read ();
  if (chr == ID) {
    mySerial.write(chr);
  }
}

}
```

Then this code is compiled and uploaded.

When finished the total sketch looks like this:

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(0, 1); // RX, TX

char ID='1';

void setup() {
  // put your setup code here, to run once:
mySerial.begin(9600);

}

void loop() {
  // put your main code here, to run repeatedly:

  if (mySerial.available()) {
  char chr = mySerial.read ();
  if (chr == ID) {
    mySerial.write(chr);
  }
```

```
}

}
```

when hitting Serial Monitor now only the number 1 should be returned in the lower window

Going back to output board – networking:

```
digitalWrite(1, HIGH);
delayMicroseconds(500); // Approximately 10% duty cycle @ 1KHz
digitalWrite(1, LOW);
delayMicroseconds(500);
```

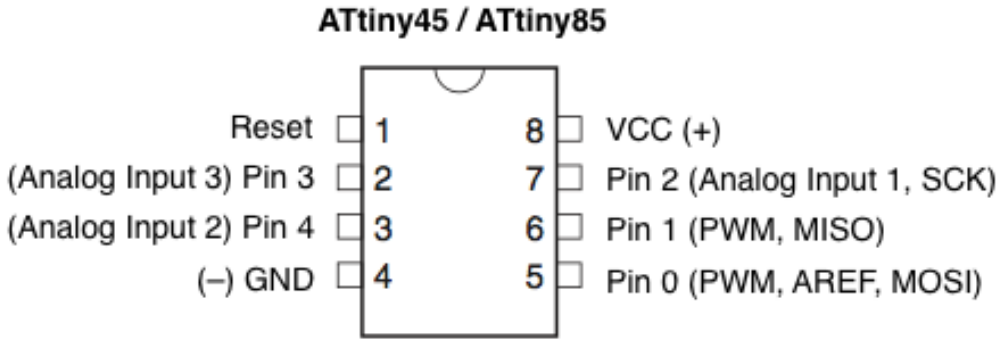
0.001 = 1 kHz
0.0001 = 10 kHz

4th May 2015

The intended networking activity is to have my output board reading light 1 and light 2.

The reading of those boards should return a value, that the output board can deliver as a sound output. It should be able to send a message to light 1, ignoring light 2 and vice a versa.

The microcontroller – pins:



Serial test:

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(0, 1); // RX, TX

char ID='1';

void setup() {
  // put your setup code here, to run once:
mySerial.begin(9600);

}

void loop() {
  // put your main code here, to run repeatedly:
```

```
  if (mySerial.available()) {
  char chr = mySerial.read ();
  if (chr == ID) {
    mySerial.write(chr);
  }
}

}
```

char ID='1' is a unique identifier – that identifies the relevant input board

Script for the reader of light (input1):

1. when you name is called read the light meter
2. then tell the reading

```
if (mySerial.available()) {  - have I heard something

char chr = mySerial.read (); - I have heard something and write it down

if (chr == ID) { - checking if the content of read is meant for him. If this is true the actor will do everything contained within the curly braces

val = analogRead(analogPin);    // read the input pin

mySerial.println(val);         // debug value – returning the message
```

Hook-up wires are connected to the input board – pin1 to TX. Nothing happens.

A line of code was taken out: pinMode(1, OUTPUT);

Opening the serial monitor – entering the number 1, returned a reading of the light.

Now the code looks like this: #include <SoftwareSerial.h>

```
SoftwareSerial mySerial(1, 2); // RX, TX

int analogPin = 3;     // potentiometer wiper (middle terminal) connected to analog pin 3

                       // outside leads to ground and +5V

int val = 0;           // variable to store the value read

char ID='2';           // variable to identify the relevant input board

int digitalPin = 1;     // potentiometer wiper (middle terminal) connected to analog pin 3


void setup() {


  // set the data rate for the SoftwareSerial port
```

```
  mySerial.begin(4800);


}

void loop() // run over and over
{



  if (mySerial.available()) {
  char chr = mySerial.read ();
  if (chr == ID) {

  val = analogRead(analogPin);    // read the input pin
  mySerial.println(val);             // debug value



   }
  }
}
```

Connecting another input board did not return any reading, when char ID='22

Script for output board – what should it do?

1. Call input1
2. Ask for reading from input1
3. Receive reading from input1
4. Respond to the intensity of light, by decreasing or increasing sound tone

```
SoftwareSerial mySerial(0, 2); // RX, TX – on input1

SoftwareSerial mySerial(2, 0); // RX, TX – on output/Master


void beep() {
   // Sound beep delay
  for (int i=0; i <= 200; i++){
  digitalWrite(1, HIGH);
  delayMicroseconds(500); // Approximately 10% duty cycle @ 1KHz
  digitalWrite(1, LOW);
  delayMicroseconds(500);
   }

   }


void lowbeep() {
   // Sound beep delay
  for (int i=0; i <= 100; i++){
  digitalWrite(1, HIGH);
  delayMicroseconds(1000); // Approximately 10% duty cycle @ 1KHz
  digitalWrite(1, LOW);
  delayMicroseconds(1000);
   }
```

```
}
```

1. instead of making beep fixed, at 1 Khz – changed it to be flexible (500+i)
2. interval between tone steps, going from interval of 1 to interval of 10

Pausing between tones

Frequency and duration can now be controlled and changed

For connecting the two boards – adding: mySerial.write('1');
This will request response from the input1

```
avrdude: initialization failed, rc=-1
         Double check connections and try again, or use -F to
override
         this check.
```

```
for (int i=0; i <= 200; i=i+10)
```

int – i=0; starts at 0 – we are going to do this as long as the i is smaller or equal to 200; how fast are we going to go.

The program was uploaded to both boards – input and output
On the output board – hit Serial Monitor to see if reading is taking place. It did.

To work with the reading of values the intervals of reading need to be defined and the speaker asked to play a certain tone, if the reading is below or beyond a certain value.

Neils code is used for the application, but sketch-networking 1 and 2 are used for networking.

Application and programming week – used Neil's code

Networking business:

Difference between '1' and 1 – for testing
'1' is represented by an number in the Ascii table = 49.
http://www.asciitable.com/
The ID that we gave the board is 49

The output board will call ID 49 and ask it to send light data.
When the microcontroller send 49 it expects to receive a light value.
The light value has a value – between 0 and 1023. It can receive those values.

The communication is limited, can only send one byte at a time.
We can only put so much info in one byte. 0-254 is the maximum value that one byte can contain. We need to split the info into packages that it can send.

Byte is made up of 8 bits. Each bit is either 1 or a 0. Lowest bit is 1 and 128 is highest. 1 – 2 – 4 – 8 – 16 - 32 ... 128

Sending 2 bytes. 3 + 255. Reconstructing... shift 3 over, 3 is first sent, the byte with the highest value is sent first. 100+30+3... is the sum of 133. 3x256 + 255.

This is a protocol, rules of how we are going to communicate.

When the output board sends 49 he input board will respond with 2 numbers, the higher value first and then the lower value.

Implementation:

First – input board:

Sending 536, first sending 5 the hundreds, dividing by 100.
We have a variable, that may or may not contain a number that is bigger than we can send in one go.

Dividing 0-1023, by 256 (8 division by 2)
An easy way to write it, by shifting. Take the whole register and shift it 8 places to the left.

Code:

```
  mySerial.write((val>>8)&&255);             // debug value (for
sending
  mySerial.write(val&&255);             // debug value
```

Sending part is now complete. Will send the high value and the low value of what it reads.

In output board:

Reconstruct the value
And make it do something – like playing two different or more tones

22.5.2015

Downloaded and added SoftwareSerialWithHalfDuplex library (remember to remove –master ending before adding the library)

Input:

In input and output networking files added:

```
#include <SoftwareSerialWithHalfDuplex.h>
//input

  SoftwareSerialWithHalfDuplex mySerial(0, 2, false, false); // RX,
TX
```

Output: #include <SoftwareSerialWithHalfDuplex.h>

```
//output

  SoftwareSerialWithHalfDuplex mySerial(2, 0, false, false); // RX,
TX
```

HIGH – LOW values

If the light is below 512 we do one thing, if it is above we do another thing

while (mySerial.available()<2); = wait until you have 2 packages

```
int highval = mySerial.read();
int lowval = mySerial.read(); = now we have unpacked the packages

val=lowval+(highval<<8); = re-stores the values

take out:
  for (int i=0; i <= 200; i=i+10){
    beep(500+i,200);


Added:

if (val<512){
      beep(1000,250);}
      else {
      beep(500,500);}
```