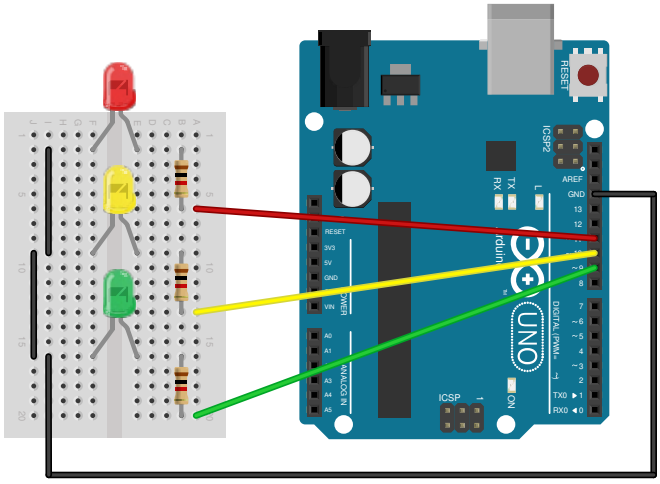
Arduino Network

Connecting Arduinos using `SoftwareSerial`

Enable native scrolling

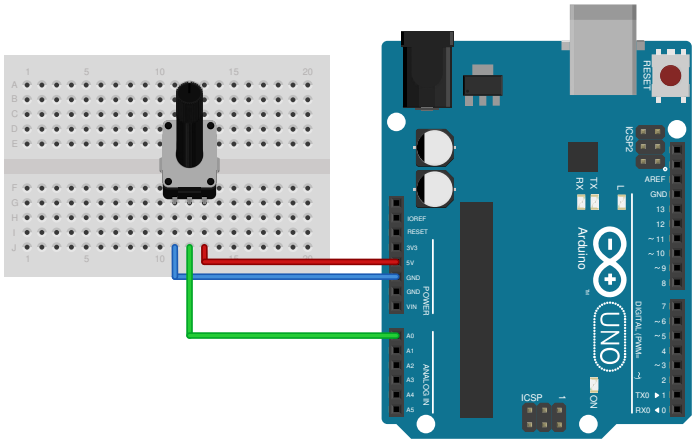
Remote status monitor



In tutorial three we used three LEDs, a potentiometer and an Arduino as a status indicator

Now we want to implement the same functionality using two Arduinos

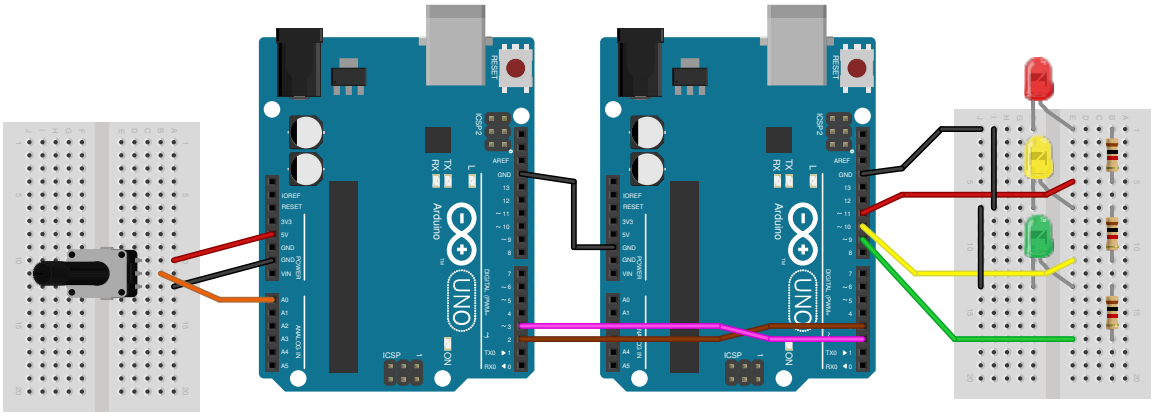
Preparation



Connect the LEDs and Potentiometer as described in tutorial three

Verify that the hardware setup works using one of the working code examples

Two Arduinos



For the next tasks we will need two Arduinos

Assemble in groups and decide who will write code for the sending / receiving unit

The Sender

The sending unit should take the analog values from the potentiometer, encode them and transfer them using `SoftwareSerial` on pins `2` and `3`

Use the code from tutorial three and the code from the lecture as a reference

Hint: You can still use `Serial.print` for debugging

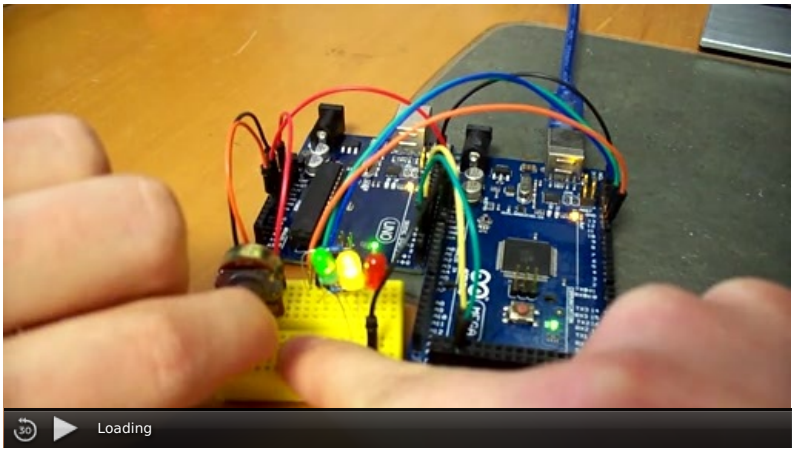
The Receiver

The receiving unit should read values using `SoftwareSerial` on pins `2` and `3`, decode them and turn on the corresponding LEDs

Use the code from tutorial three and the code from the lecture as a reference

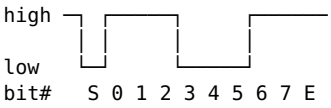
Hint: You can still use `Serial.print` for debugging

Working Example



Sender / Receiver

DIY Serial.write (optional)

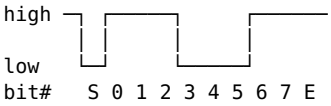


The baud rate is the number of symbols that are transmitted per second

A symbol can be the start bit, a data bit, a stop bit or a parity bit

The defaults are 9600Bd, one start bit, eight data bits, no parity bit and one stop bit

DIY Serial.write



Write a sender that uses `digitalWrite` and `delayMicroseconds` to transfer the data

Hint: The following slides contain more hints should you get stuck

Hint #1

```
1 pinMode(3, OUTPUT);
2 digitalWrite(3, s);
```

The software serial port used pin 3 as `TX` line,

this is where data should be sent

Hint #2

```
void uart_bit_send(boolean s)
{
  digitalWrite(3, s);
  delayMicroseconds(104);
}
```

9600 Symbols are transmitted per second,
this means that the transmission of one symbol takes 1s/9600 ≈ 104µs

Hint #3:

```
uart_bit_send(LOW);
// Transfer data
uart_bit_send(HIGH);
```

The transmissions start with a 0 (start bit) and ends with a 1 (stop bit)

Working example

```
uart_bit_send(LOW);
for(uint8_t b=0; b<8; b++) {
  uart_bit_send(defcon & 0x01);
  defcon>>=1;
}
uart_bit_send(HIGH);
```

...