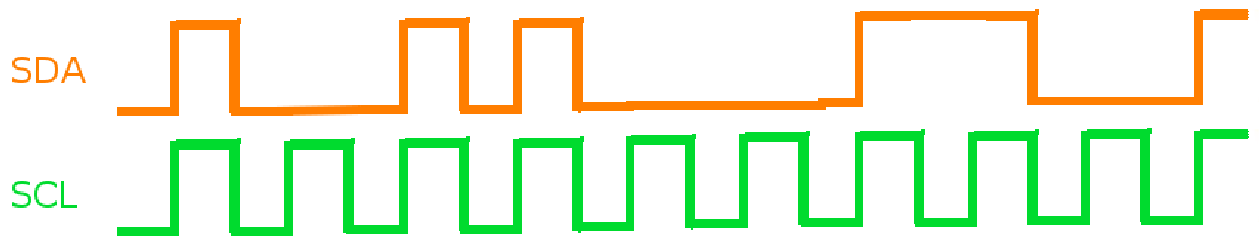
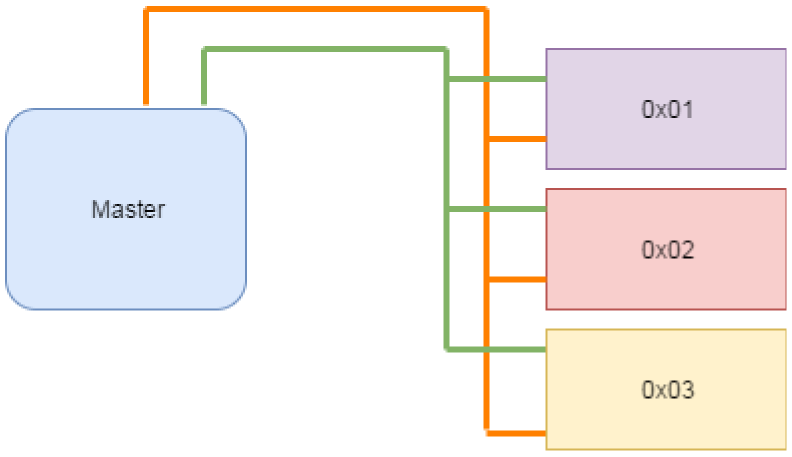
The I2C Bus

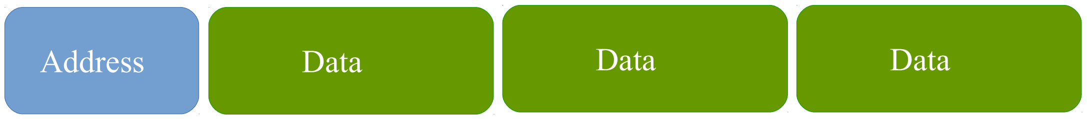
Inter Integrated Circuit or I2C is a very popular interface for communicating between devices. There are only two wires involved in the I2C interface, the clock line (SCL) and the data line (SDA). The I2C interface requires that both devices be tightly synchronized. The clock line's purpose is to ensure this level of synchronization. The clock signal varies in a very regular fashion, and whenever it changes the device knows to read a new bit from the data line. You can see from the image below how this allows the device to read a new bit from the line even when the bit did not change from the previous value, like sending two zero's in a row.



If the I2C interface was only designed to facilitate communication between two devices this might be all we need; but the I2C is designed to facilitate communication between potentially over a thousand different devices. The question arises of how do we specify which device we are intending to communicate with. This is where the addresses we've seen in our sketches come into play. Each device on the I2C bus has a unique address that's hard coded to the device.



When data is written to the I2C bus, it is done so with a very specific structure. First, we write a 7-bit address. Any devices with an address that do NOT match this 7-bit address will ignore the rest of the data. After sending the address, we can send some number of standard 8-bit bytes to the device.



Later, we can specify a different address and send some commands and receive some replies from another device. Like all serial buses, I2C does not allow us to communicate with multiple devices simultaneously; its one device on the line at a time. Since we only have the one data line, it needs to be shared for both communications from the master to the device, and the device to the master.

The I2C interface provides us a way to read and write bytes to and from a serial device. What it does NOT do is provide any kind of standard for what those bytes *mean.* For instance, one some example device, a pattern of 1001 might be the ‘magic word’ that asks the device to reset; while the pattern 0110 might ask it to return some measurement. I2C is completely agnostic about all of this. This information is contained in the library (built on *top* of the I2C interface) that allows the microcontroller to inter-operate with the device. You can think of the infrastructure I2C provides as being like a telephone line between devices, but once that line is established you still have to know whether to speak English, Spanish or Japaneses to the person on the other end if you want to get anything done.