COMP140 GAM160: Hacking Hardware/Advanced Programming

Session 9: 12

Learning outcomes

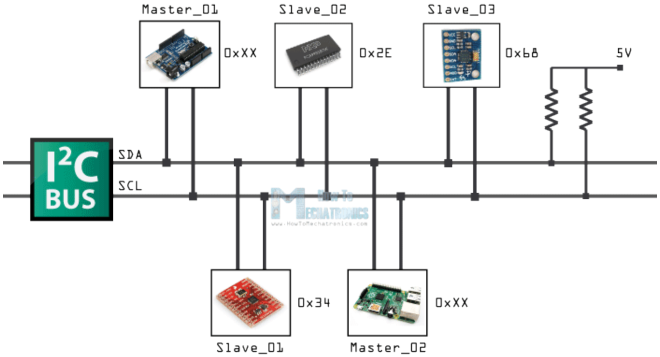
- **Identify** the various protocols used to communicate with devices attached to the Arduino
- **Explain** the difference between asynchronous & synchronous communication
- **Outline** the strengths and weaknesses of each protocol

Protocols

The main protocols used to communicate with devices from the Arduino

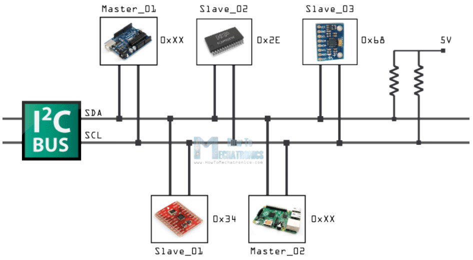
- **I2C** - The Inter-integrated Circuit (I2C) Protocol
- **SPI** - Serial Peripheral Interface
- **Dallas one-wire** - microLan

I2C Bus



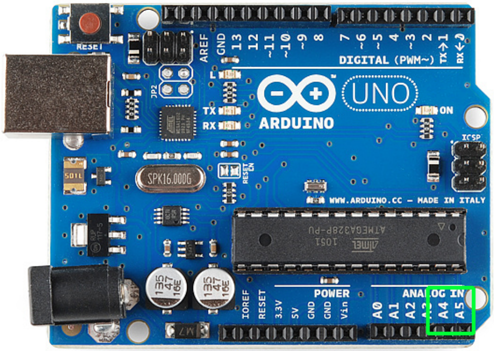
SCL is the clock line. It is used to synchronize all data transfers over the I2C bus. SDA is the data line. The SCL & SDA lines are connected to all devices on the I2C bus

I2C Bus



There should be a resistor from the SCL line to the 5v line and another from the SDA line to the 5v line. You only need one set of pull-up resistors for the whole I2C bus, not for each device. The value of the resistors is not critical. 1k8, 4k7 and 10k are common values.

I2C Bus

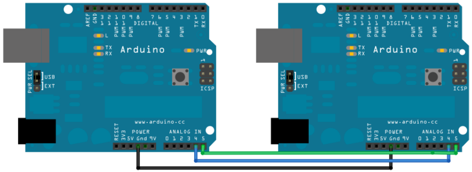


Uno uses pins A4 for SDA and A5 (SCL). I2C is easy to setup using the Wire Library preinstalled in the Arduino IDE.

Master & Slaves

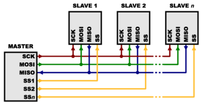
The devices on the I2C bus are either masters or slaves. The master is always the device that drives the SCL clock line. The slaves are the devices that respond to the master. A slave cannot initiate a transfer over the I2C bus, only a master can do that. There can be, and usually are, multiple slaves on the I2C bus, however there is normally only one master.

I2C Experiment



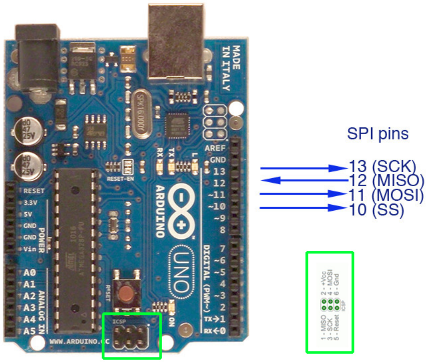
https://www.arduino.cc/en/Tutorial/MasterWriter

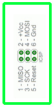
SPI Bus



- **MISO** - (Master In Slave Out) - The Slave line for sending data to the master
- **MOSI** - (Master Out Slave In) - The Master line for sending data to the peripherals,
- **SCK** - (Serial Clock) - The clock pulses which synchronize data transmission generated by the master
- **SS** - (Slave Select) - the pin on each device that the master can use to enable and disable specific devices.

SPI Bus





SPI is easy to setup using the SPI Library preinstalled in the Arduino IDE.

Master & Slaves

In general, each slave will need a separate SS line. To talk to a particular slave, you?ll make that slave?s SS line low and keep the rest of them high (you don?t want two slaves activated at the same time, or they may both try to talk on the same MISO line resulting in garbled data). Lots of slaves will require lots of SS lines; if you?re running low on outputs, there are binary decoder chips that can multiply your SS outputs.

Asynchronous - SPI

Asynchronous means that data is transferred without support from an external clock signal. This transmission method is perfect for minimizing the required wires and I/O pins, but it does mean we need to put some extra effort into reliably transferring and receiving data.

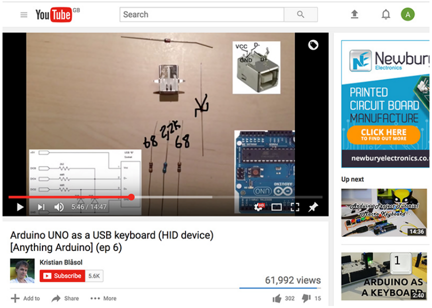
Synchronous - SPI

A **synchronous** serial interface always pairs its data line(s) with a clock signal, so all devices on a synchronous serial bus share a common clock. This makes for a more straightforward, often faster serial transfer, but it also requires at least one extra wire between communicating devices. Examples of synchronous interfaces include SPI, and I2C.

Dallas One-wire & NeoPixels

Other protocols to watch for!

Emulate USB Keyboard



Arduino UNO as a USB keyboard (HID device) [Anything Arduino] (ep 6)

Kristian Blåsol

61,992 views

https://www.youtube.com/watch?v=RoG_-9lAnSI