

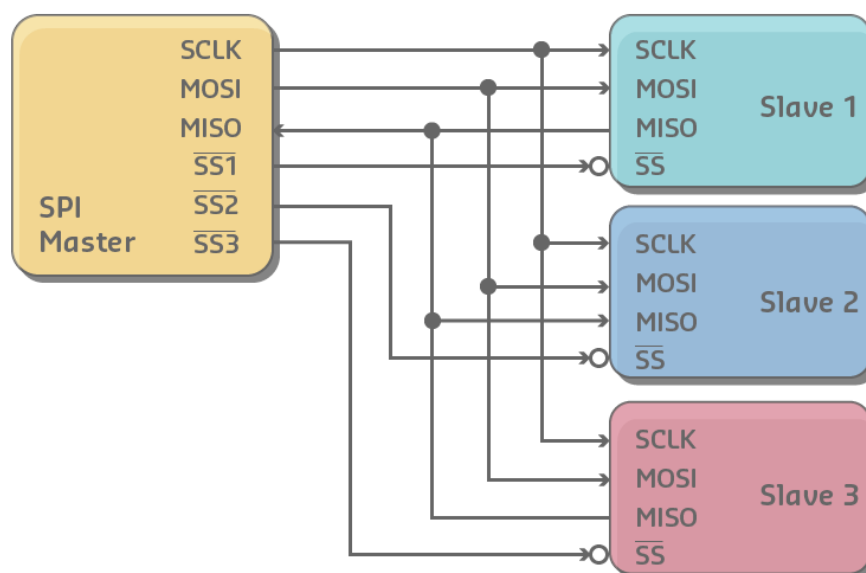
## SPI Bus

*Buses are used for wired communication between devices over relatively short distances. Serial buses send all bits of data down a shared line one after the other. This contrasts with parallel buses, which can send many bits at the same time but require a separate physical line for each bit. This creates some timing problems and other limitations. Serial buses reduce the number of physical wires needed. There are many different formats and protocols used for serial communications.*

In this activity we will look at a simulation of communication between two devices using one type of serial bus called Serial Peripheral Interface (SPI).

SPI requires only four physical connections if communication is between two devices. One device will act as the **master** and one as the **slave**. See the four connections between **SPI Master** and **Slave 1** in the diagram below.

Extra slaves can be added to the bus by adding only one new connection to the master for each, as can be seen in the connections to **Slave 2** and **Slave 3** in the diagram below.



SPI can operate in **full duplex** mode, which means communications between a master and slave can be sent in both directions at the same time.

It is also a **synchronous** form of serial communications, meaning it requires use of a shared clock for synchronising data sending and receiving between devices.

In SPI, there is no specification for the protocol used between devices so that message sizes and data rates may vary as required for the particular application. Acknowledgment of transmission by the receiver is not required either but can be built in if it is seen as useful.

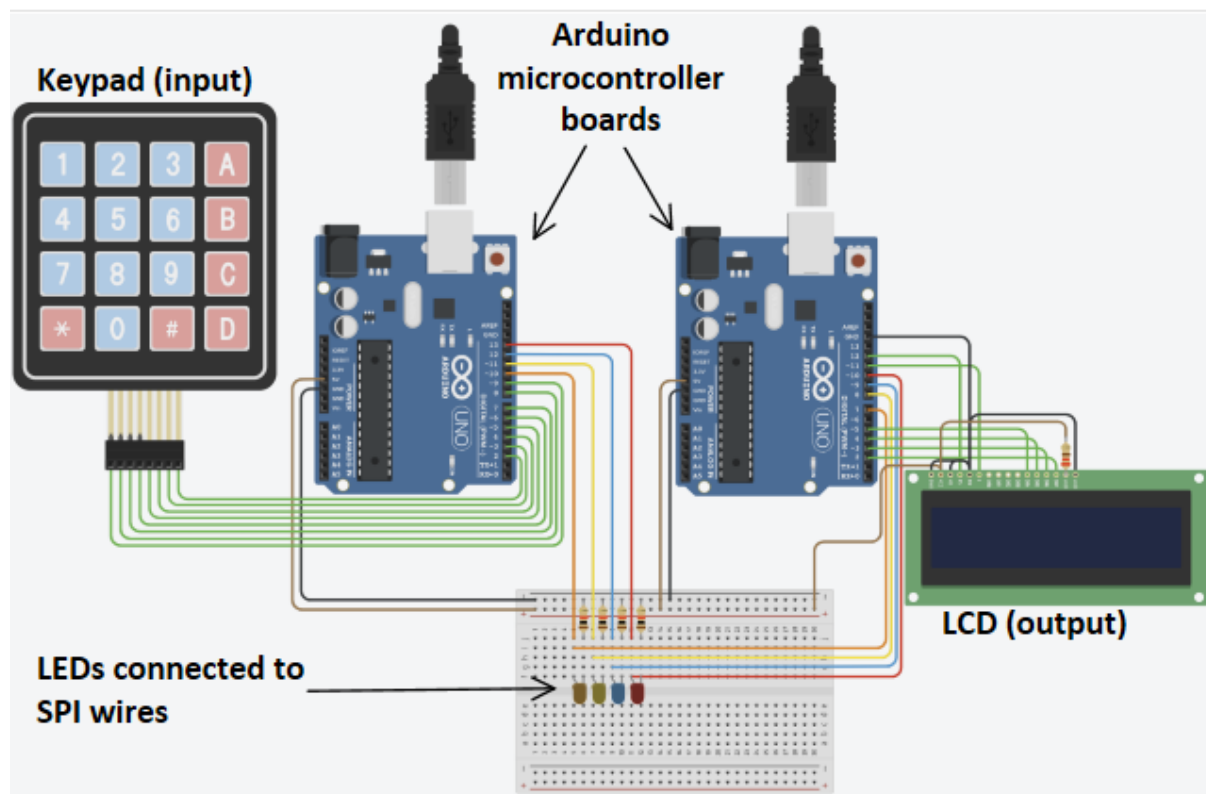
**SCLK** - Serial Clock. This is the clock signal being sent by the master to any slaves. Devices will use the moment of either the rising or falling edge (low-to-high or high-to-low) of the

clock line to read data and use the opposite edge to make any changes to data on the line they are sending.

**MOSI** - Master Out, Slave In. This line is for sending data from master to a slave. The master switches the voltage to its high or low level before the next clock edge to represent a '1' or '0'.

**MISO** - Master In, Slave Out. This line is for sending data from a slave to the master. The slave switches the voltage to its high or low level before the next clock edge to represent a '1' or '0'.

**SS** - Slave Select. There is a separate SS line between a master and each slave. These lines are normally held high but are lowered to indicate when a data line should be written to or read from.



"Autodesk screen shots reprinted courtesy of Autodesk, Inc."

The picture above is a screen capture of a circuit which connects a keypad, a liquid crystal display (LCD), 2 Arduino microcontroller boards and 4 light emitting diodes (LEDs) with current limiting resistors.

**Follow the instructions below to open and use this simulation:**

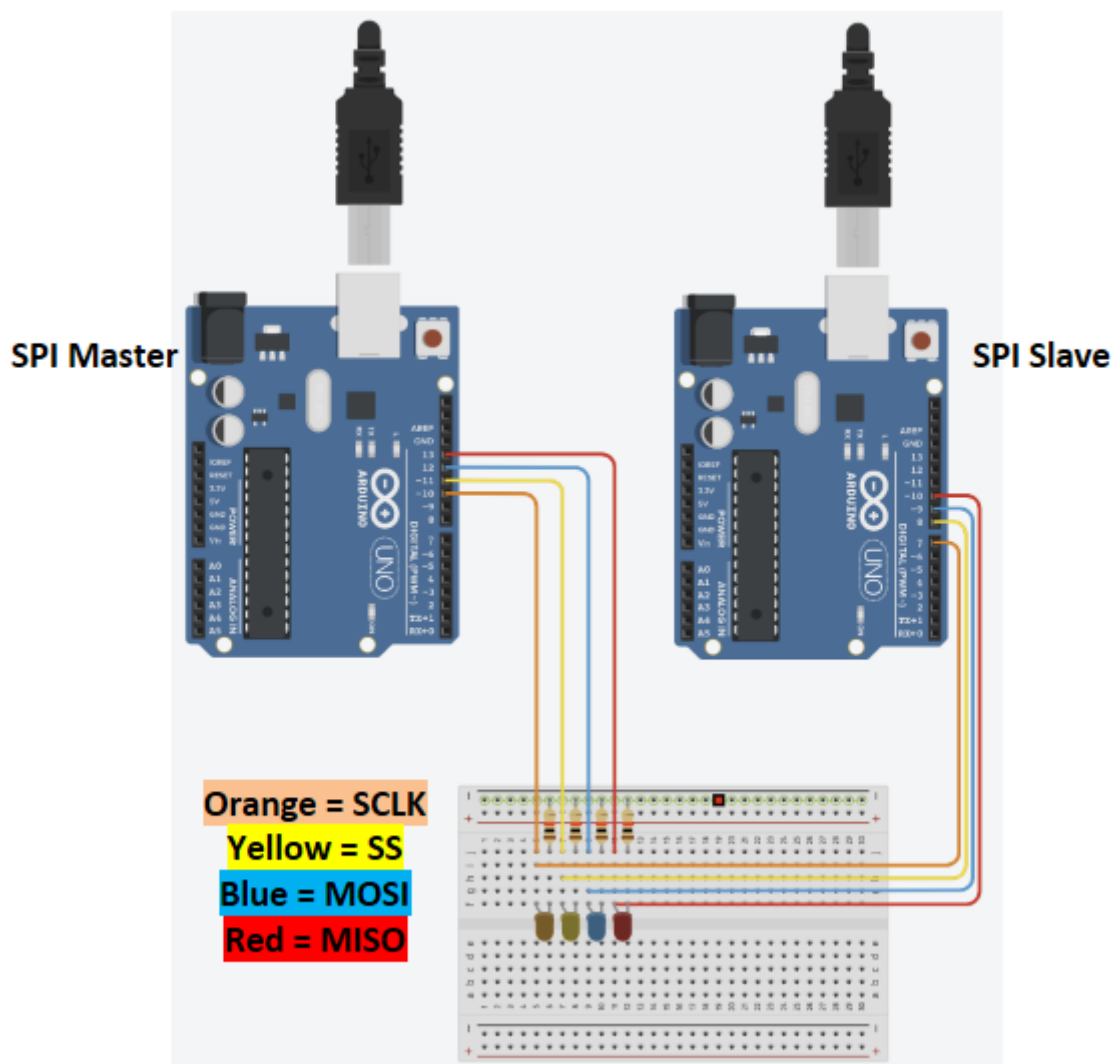
1. Log in to your Tinkercad® account. It works best using the Google Chrome browser. If you are not familiar with how to use Tinkercad® Circuits then watch the **Introduction to Tinkercad®** video.

2. Once logged in, click this link <https://www.tinkercad.com/things/arTQo0KMOQf> which will open a preview of the **SPI Bus** circuit.
3. Click the **Copy & Tinker** button.  
This will save a copy of the circuit to your account and open the simulation for you to use. It will be automatically named **Copy of SPI Bus**.

Copy & Tinker

If you are not familiar with electrical circuits the wiring may seem complicated at first glance. For this activity we can ignore the green wires which connect the keypad and LCD to the Arduino boards. We can also ignore all brown and black wires which are for the 5V power supply and ground connections.

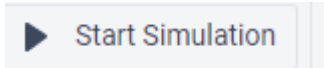
The picture below has been simplified by removing all components and wires other than the four lines which form the SPI bus connection for communication between the two Arduino boards. These lines have been labelled in the picture.



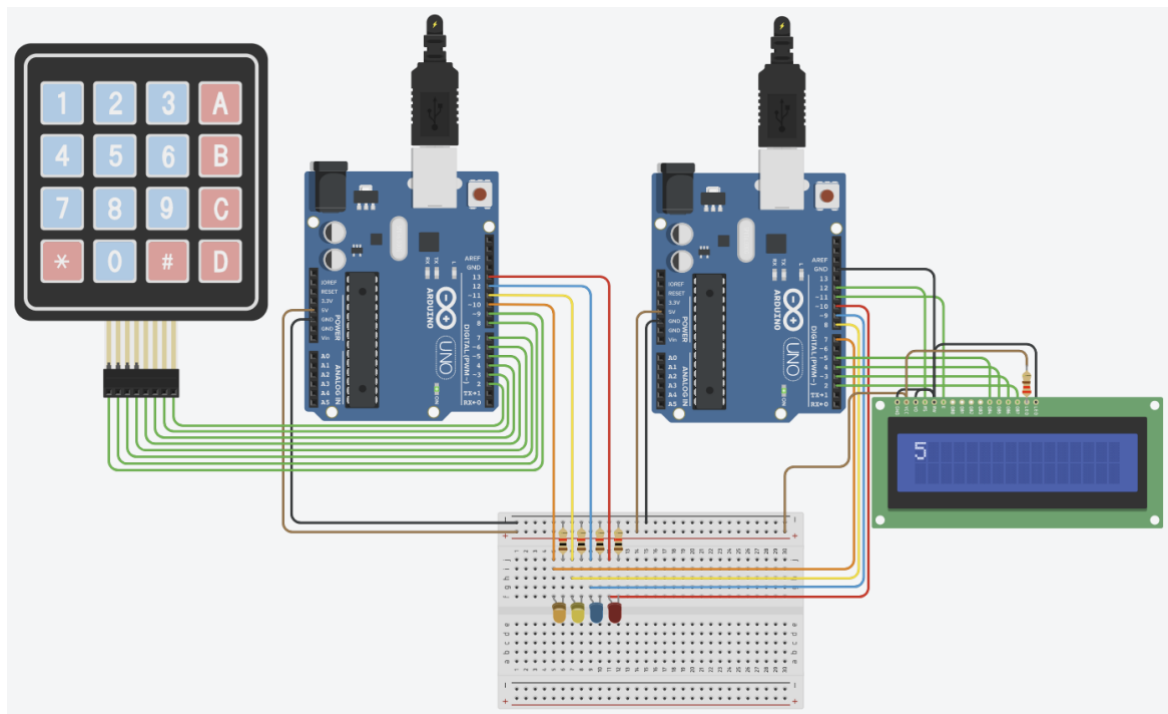
"Autodesk screen shots reprinted courtesy of Autodesk, Inc."

For this activity the LEDs have been connected to the lines to give a visual representation of whether the lines are on or off. An LED will be switched on when the signal on the line represents a '1' and off when the signal represents a '0'.

Follow the steps below to send data across the SPI bus while viewing the transmission:

1. Click the **Start Simulation** button, which is towards the top-right of your Tinkercad® workspace. 
2. After initialization, you will see the Arduinos power up. Look at the orange LED for the SCLK (Serial Clock) line. You will see it being changed between on and off once per second by the SPI Master. The SPI Slave monitors this line for timing. A real SPI bus would have a much higher clock rate but this simulation has been slowed down so you can see the transmission take place.
3. Notice that the yellow SS (Slave Select) line is always on. This is kept high by the SPI Master until it is ready to send to or receive from the SPI Slave.
4. Click button 5 on the keypad and watch the transmission process on the LEDs. The process is described below.
  - You will soon see the yellow SS LED turn off to indicate to the slave that transmission is beginning.
  - While the SS light is off, every time the orange SCLK line transitions from off to on, the SPI Slave will take note of whether the blue MOSI (Master Out Slave In) line is off or on at that moment.
  - For each clock transition, it will record MOSI off as a '0' bit and MOSI on as a '1' bit until it has received all 4 bits.
  - In this example, the SPI Slave will see 0101 on the MOSI line, which is a binary representation of the number 5.
  - When the SPI Slave has received 4 bits, it will briefly switch on the MISO (Master In Slave Out) line as message of acknowledgement to the master.
  - The SPI Slave will now display the message on the connected LCD, where you will see the number 5.
  - Once the bits have been sent and acknowledgment has been received, the SPI Master will switch the yellow SS line on again. The SPI Slave can now ignore any changes on all other lines.

In the screenshot below, the number 5 has been successfully sent from one Arduino to the other via the SPI bus.



"Autodesk screen shots reprinted courtesy of Autodesk, Inc."

Try different buttons on the keypad and watch the process as the data is sent over the SPI lines. The table below has been provided as a reference for the binary representation of each key and how it is shown on the LCD.

| Key | Binary | Display |
|-----|--------|---------|
| 0   | 0000   | 0       |
| 1   | 0001   | 1       |
| 2   | 0010   | 2       |
| 3   | 0011   | 3       |
| 4   | 0100   | 4       |
| 5   | 0101   | 5       |
| 6   | 0110   | 6       |
| 7   | 0111   | 7       |

| Key | Binary | Display |
|-----|--------|---------|
| 8   | 1000   | 8       |
| 9   | 1001   | 9       |
| A   | 1010   | 10      |
| B   | 1011   | 11      |
| C   | 1100   | 12      |
| D   | 1101   | 13      |
| *   | 1110   | 14      |
| #   | 1111   | 15      |

#### Some differences in the simulation compared with a real SPI bus:

- This simulation has been slowed to a rate which is visible to us. If we were viewing a real transmission it is likely to appear as though instant. The actual clock rate set by the master may vary in different SPI systems provided all slaves are capable of keeping up with the rate.

- No acknowledgment of transmission is required. For this reason, a master may transmit and simply assume a slave is connected and ready to receive. It's possible that no slave is connected at all. An acknowledgment method can be designed which may be similar to what is used here if there is a need for it.
- This simulation transmits only 4 bits at a time, which is half a byte (called a nibble). A real SPI transmission is likely to contain many more bits. There is no standard for length but all components on an SPI bus should understand what length to expect on their bus.

**You have now completed the SPI Bus activity. See the RS232 Tinkercad® activity for a different method of communication between devices.**

*“Autodesk, AutoCAD, Tinkercad®, DWG, the DWG logo, and Inventor are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and other countries.”*