LABORATORY MANUAL

Cx1106
Computer Organization and Architecture

Lab Experiment #3

Arduino Development Platform

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY

1. OBJECTIVES

- Develop Arduino Sketch programs
- Exchange data between two devices using UART Serial communication.
- Exchange data between two devices using Synchronous Serial communication.
- Interface Arduino to various input devices e.g. Buttons, Ultrasonic sensor, Photo cell.
- Interface Arduino to various output devices e.g. LED, Buzzer.
- Capture and analyze the data using an oscilloscope.

2. EQUIPMENT

- A Windows-based computer (PC) with a Universal Serial Bus (USB) port.
- Arduino UNO R3 board.
- Breadboard and hardware components
- USB cat A-B cable.
- Oscilloscope.
- Please do not remove any pre-connected wires from the breadboard after your experiment.

3. LITERATURE READING

- Week8a lectures on
  - Computer Organisation Overview
  - Signal Chain Sub-System
- Arduino Development Platform. Some online resources
  - https://www.arduino.cc/en/Guide/HomePage
  - https://www.makerspaces.com/arduino-uno-tutorial-beginners/

4. INTRODUCTION

Arduino is a popular development platform in the maker's community. In this lab, you will implement some simple projects on this platform and in the process, be introduced to computer interface and signal chain related topics. The lab exercises are adapted from sample tutorials supplied by LAFVIN.

5. Arduino Development System

Fig. 1 shows an Arduino UNO R3 board that is based on the Atmel ATmega328 processor (See Red Square). The pins on the processor is extended out to headers on each side of the board and labelled accordingly, e.g. Pin 0 is the UART RX pin and Pin1 is the UART TX pin (See Blue Square).

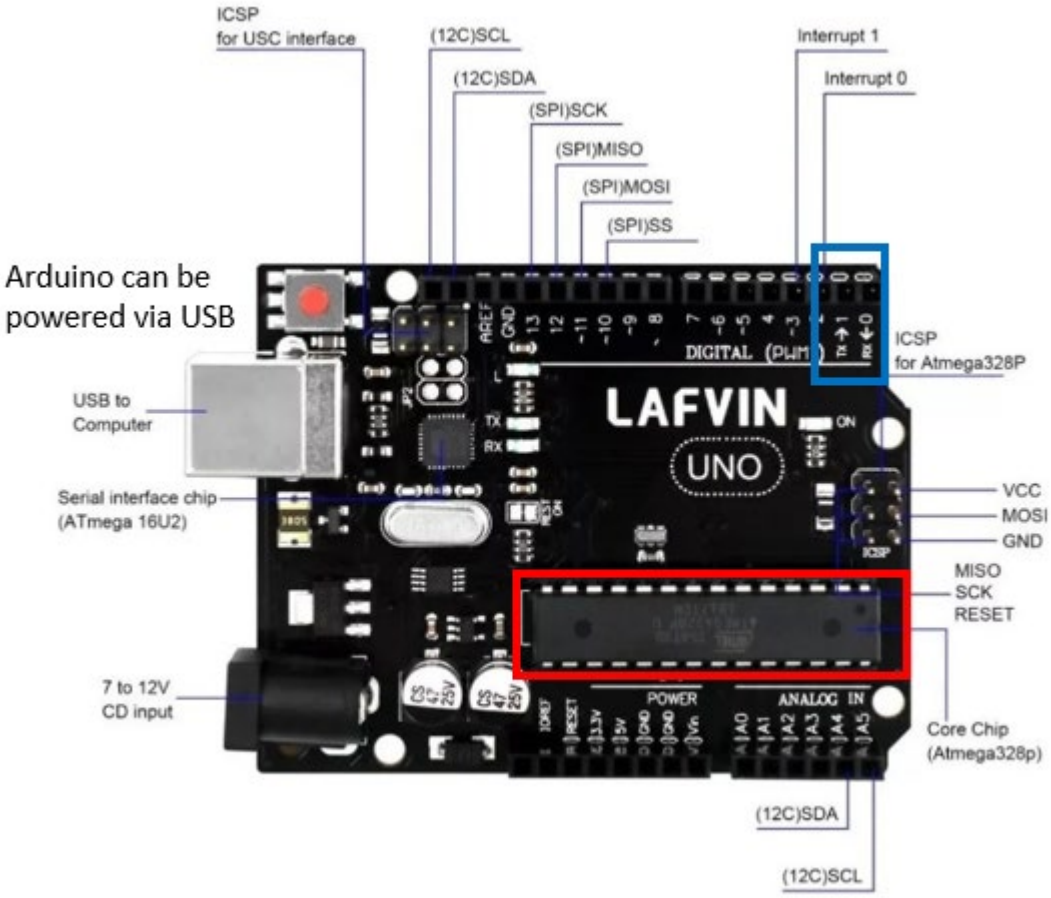


Fig. 1: Arduino UNO R3

5.1 Pin description

Some important pins that you will be using in the lab

Table 1: (Some) Pin description

GND	Ground Pin. This is the Zero Volt reference for the Arduino Board.
Digital I/O Pins	Labelled as Pin 0-13 in the row of headers nearer to the USB port. Note that the same pin can be used for other functions as well, e.g. Digital Pin 0 is also the UART RX (Receive) pin.
3.3V	Supplies 3.3V output to external devices.
5V	Supplies 5V output to external devices.
Analog pins	Labelled as A0-A5 on the row of headers further from USB port. Connects to the Analog-to-Digital Converter on the processor.

5.2 Breadboard

A breadboard is used to make up temporary circuits for testing or to try out an idea. No soldering is required so it is easy to change connections and replace components. Parts will not be damaged so they will be available to re-use afterwards.

Fig. 2 shows how the breadboard holes are connected. The top and bottom rows are linked horizontally all the way across. The power supply is connected to these rows, + at the top and 0V (zero volts) at the bottom. On larger breadboards there may be a break halfway along the top and bottom power supply rows. It is a good idea to link across the gap before you start to build a circuit, otherwise you may forget and part of your circuit will have no power! This has been done for the breadboard you are using in this lab.

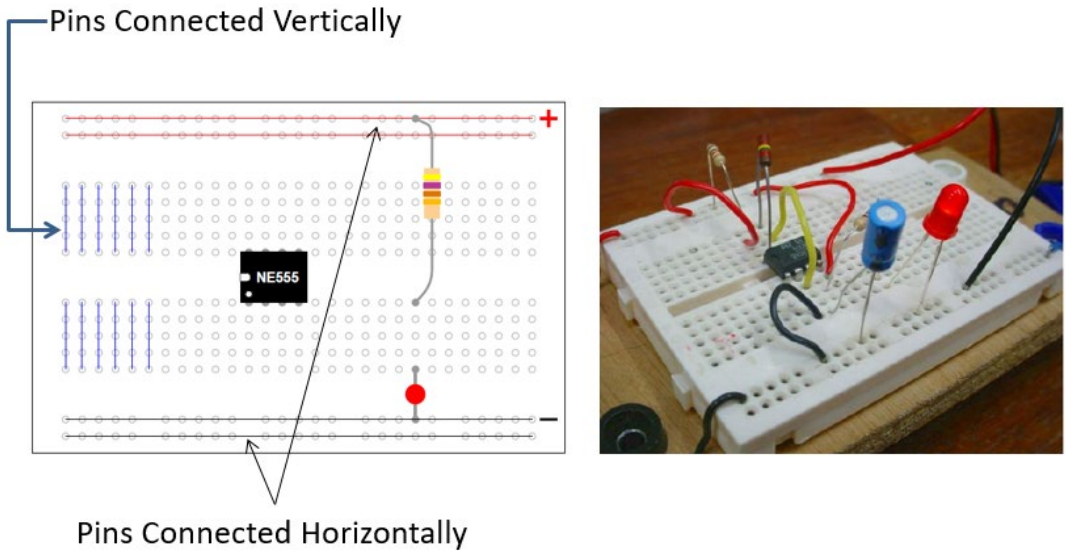


Fig. 2: Breadboard

5.3 Connecting the Arduino UNO to the PC

Connect the USB cable to the Arduino board and the PC. You should see the green LED "ON" light up, indicating power is applied to the Arduino board via the USB cable. There is no need to provide additional external power source. On the PC, open the START menu and search for ***Device Manager***. Scroll to ***Ports (COM & LPT)***, open the tab and determine the assigned COM port for the Arduino board. Fig. 3 illustrates an example that assigns COM5 to the Arduino UNO board. The actual COM port number may differ on your PC.

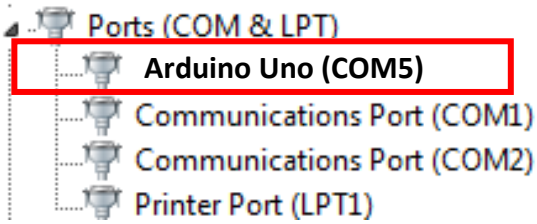


Fig. 3: Assigned Arduino COM Port

Double-click the Arduino icon on the PC desktop to start the Arduino software. Select ***Tools*** >> ***Board*** >> ***Arduino Uno***. Fig. 4 illustrates the selection. **It is important that you select the correct development board.**

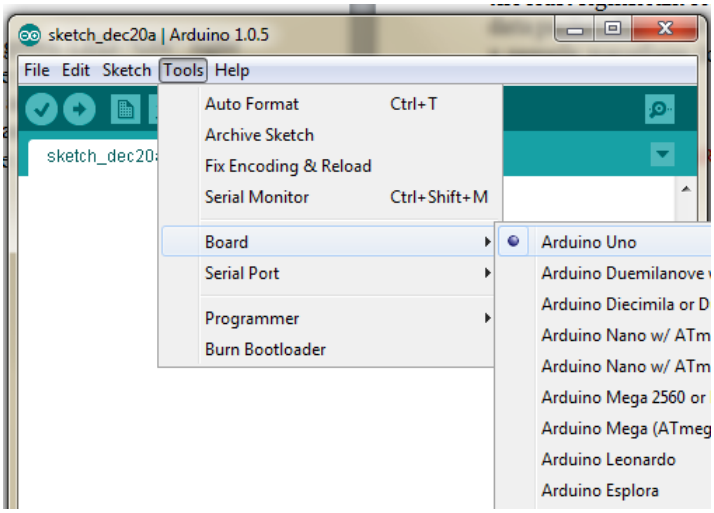


Fig. 4: Select Arduino Uno in the Arduino Software

Next, select ***Tools*** >> ***Serial Port*** >> ***COM5***. Fig. 5 illustrates an example. **The PC can connect to the Arduino board only if the serial port is set correctly.**

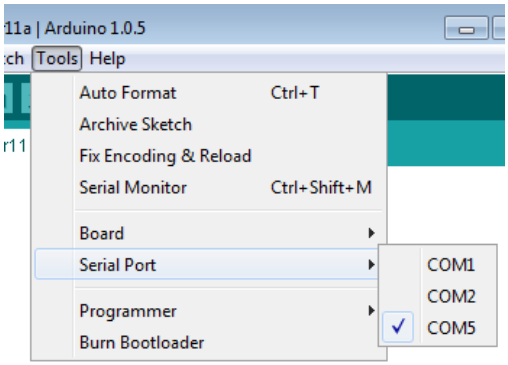


Fig. 5: Select Assigned COM Port

5.4 Basic Arduino Sketch

A typical Arduino program is called a sketch and is made up of two routines.

- The routine ***setup()*** is run once every time the Arduino powered up. The routine is used for resource allocation such as configuring the pins.
- The routine ***loop()*** is the main program and is executed repeated in an infinite loop.

Fig. 6 illustrates a simple sketch where ***pin 8*** is assigned to variable ***led*** in the routine ***setup()***. The routine ***loop()*** is the main program where ***pin 8*** is assigned to logic "***HIGH***".

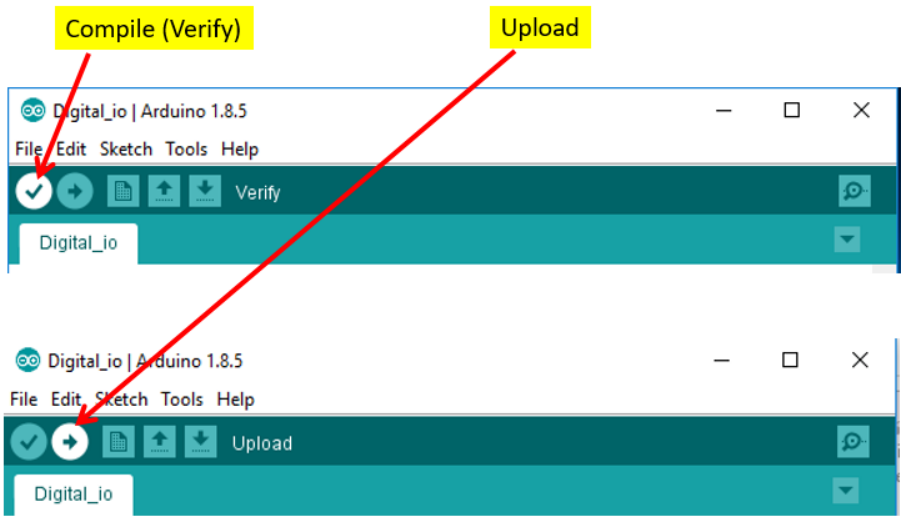
```
int led = 8;      // the pin that the LED is attached to

void setup() {
  // declare pin 8 to be an output:
  pinMode(led, OUTPUT);
}

void loop() {
  digitalWrite(led, HIGH);
}
```

Fig. 6: Basic Arduino Sketch

5.5 Compiling and Downloading Arduino code

After writing the code, you can compile and upload the code by clicking the buttons show in the figure below.

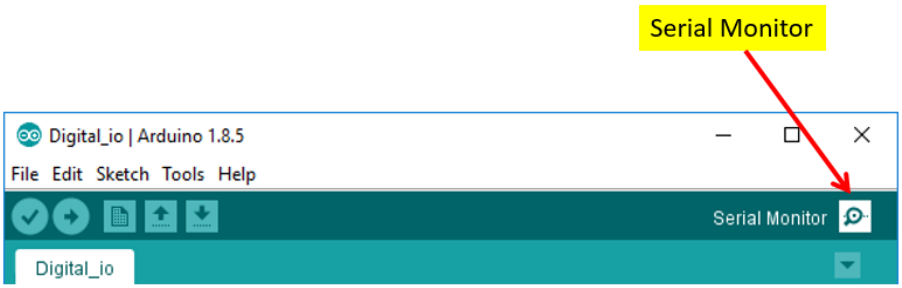
Compile/Verify button will compile the code written into machine code that the Arduino processor understand and use but the machine code generated is still resided on the PC. Upload button will transfer the machine code to the Arduino processor on the board via USB. The code you wrote in the Arduino Sketch Application is meant for execution on the Arduino Processor.



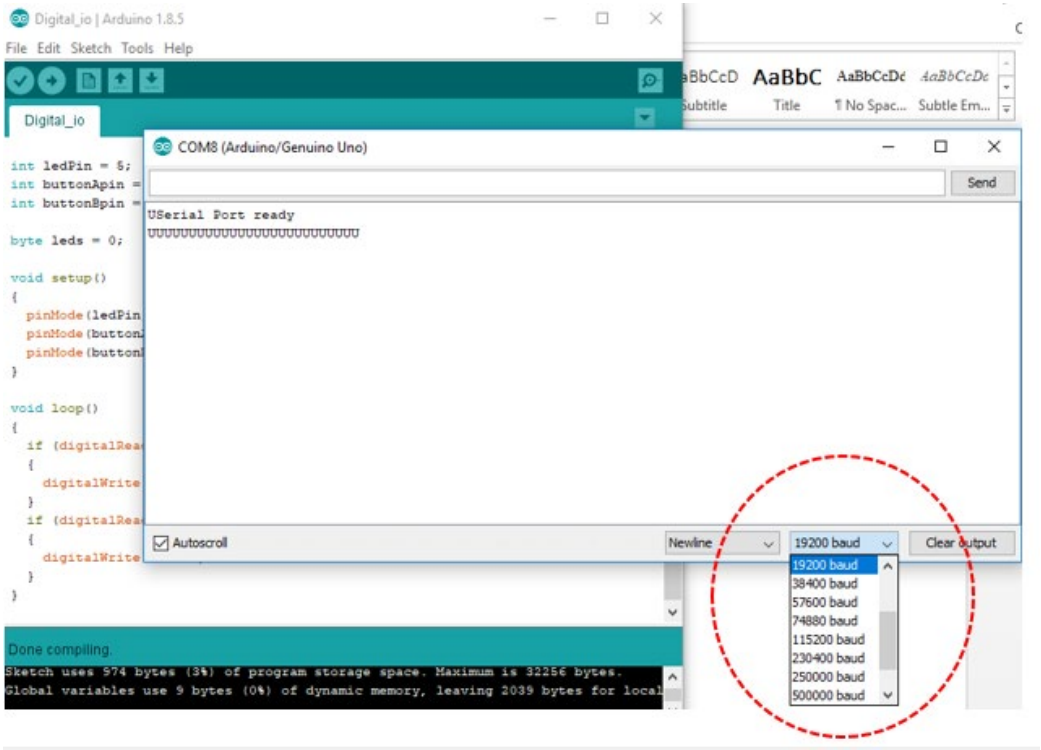
5.6 Serial Monitor

Serial Monitor is a very useful tool that allows user to view status of the code execution. The serial monitor display the information transmitted from the Arduino Processor's hardware UART peripheral. This allows user to feedback status of key variables that tracks the progress of the code execution.

To bring up the Serial Monitor, click on the button shown in the figure below



To be able to display information correctly, the baud rate has to be the same as that configured in the Arduino code. Baud rate used by the serial monitor is decode the received information can be configured from the dropdown menu shown in the figure below.



5.7 ASCII Table

The ASCII table requires 128 data representations. Table 2 shows the ASCII representations of upper-case and lower-case alphabets in hexadecimal values. Hence, to send an ASCII character, at least 7 data bits are needed.

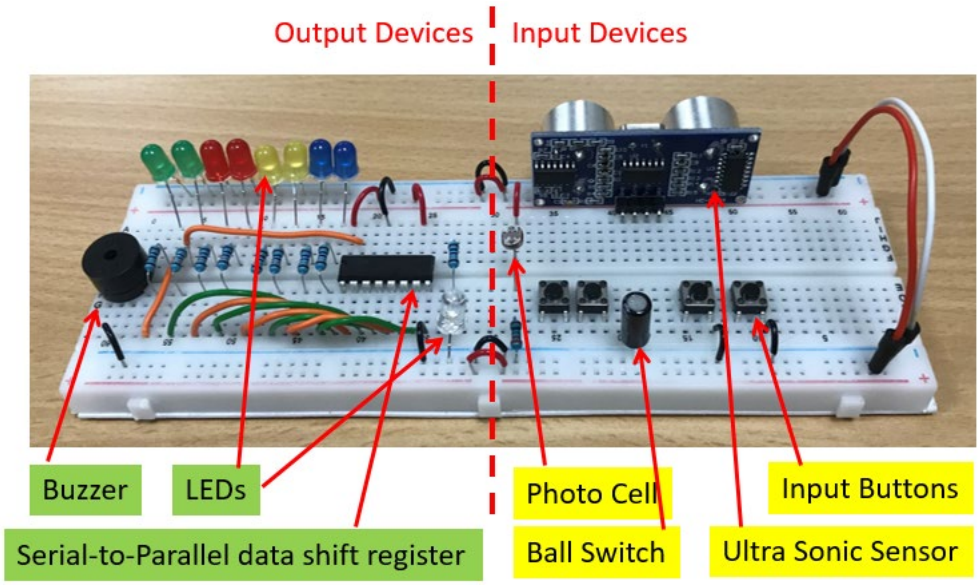
Table 2: ASCII Table

ASCII (HEX)	Char (Upper)	ASCII (HEX)	Char (Upper)	ASCII (HEX)	Char (Lower)	ASCII (HEX)	Char (Lower)
41	A	4E	N	61	a	6E	n
42	B	4F	O	62	b	6F	o
43	C	50	P	63	c	70	p
44	D	51	Q	64	d	71	q
45	E	52	R	65	e	72	r
46	F	53	S	66	f	73	s
47	G	54	T	67	g	74	t
48	H	55	U	68	h	75	u
49	I	56	V	69	i	76	v
4A	J	57	W	6A	j	77	w
4B	K	58	X	6B	k	78	x
4C	L	59	Y	6C	l	79	y
4D	M	5A	Z	6D	m	7A	z

6. EXPERIMENT

6.1 Hardware Overview

The Arduino will be used to communicate with various input/output devices on a breadboard in this lab. Figure below shows an overview of the breadboard with the various devices. Please do not remove any pre-connected wires from the breadboard after your experiment.
Note that some of the devices will only be used in optional sections of the lab. Questions related to specific exercise done in optional sections will NOT appear in the lab quiz nor the lab handout.



6.2 Oscilloscope

An oscilloscope is an equipment used to probe and display electrical signals. The vertical axis is the voltage level detected while the horizontal axis is the time in which the voltage is sampled. For your lab, the Agilent DSO6014A model (or similar model) is used. A pictorial of the front panel is shown in Fig.7 with some of the key buttons/knobs labelled. If a different scope is used, the layout of the keys will be different but their function will be similar to those described below. You'll revisit this section again when you need to use the scope to test for DC voltage and to probe the UART TX signal.

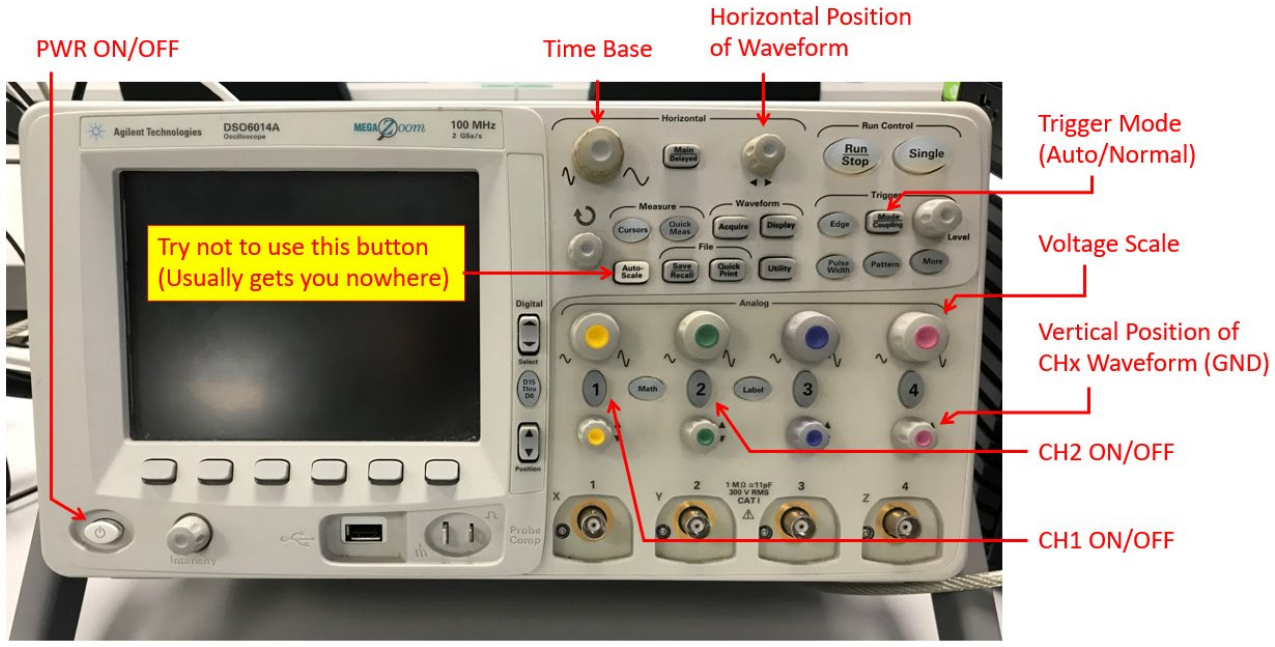


Fig.7: Front panel of Agilent DSO6014A Oscilloscope

Description of key functions

(a) **Power button** turn on the Oscilloscope (hardware isn't that difficult correct? ☺).

(b) **Time Base.** This controls the display scale of the time axis (horizontal axis). It goes by time/div, where 'div' refers to one square width on the scope. e.g. 100μs/div means each square width represent 100μs. To allow a full UART packet to be displayed on the screen, adjust the "Time/Div" knob to around 100μs/div.

(c) **Horizontal Position.** Turning this knob moves the captured waveform horizontally on the display.

(d) **Trigger Mode.** This controls how the scope synchronise the capturing of waveforms. Typically, we use 'Auto' mode when capturing DC waveform (signals that don't change with time), and 'Normal' mode to capture time varying waveform such as UART transmission signal.

(e) **Voltage Scale.** Controls the display scale of the voltage axis (vertical axis). Uses volts/div configuration similar to the time/div used in Time Base adjustment. Voltage/div should be around 1 or 2V for this experiment.

(f) **Vertical Position.** Adjust vertical position of the waveforms. Useful when trying to position two waveforms (vertically) on the display.

(g) **Channel ON/OFF.** Toggle button to enable/disable the display of the waveform captured from Ch1/Ch2 scope input.

(h) **Trigger Source.** When in 'Normal' Trigger Mode, you need to select the corresponding channel you are using as the trigger source. Each scope can only trigger on one input channel at any instance. Check with you friendly supervisor or lab executives on how to select the trigger source on the scope.

6.3 Digital input and output to Arduino Board

In this section, we will use two buttons to turn an LED ON/OFF. A program is written to read the logic level reported at the pins connected to the buttons. A logic '1' or a logic '0' will be output to the pin connected to the LED depending on which of the two buttons is pressed.

6.3.1 Hardware

#Task: Make the following connection between the Arduino and the Breadboard

- Pin 8 and Pin 9 of processor to button B1 and B2 respectively (the legs not connected to ground).
- Pin 5 of Arduino to the Positive Terminal of the LED (the leg not connected to ground).
- See Fig 9 for the connections. Note that if there are no separate power supply to the breadboard, you need to draw the power from Arduino Board's +5V. The photo in Fig.9 shows the +5V and Ground headers of Arduino board connected to the breadboard (the red and white wire from Arduino board).
- The photo in Fig.9 also shows how the oscilloscope probe needs to be connected to detect the voltage level at input pin of the processor.

The buttons used has four legs, two legs are connected internally to each other, pressing the button will cause all four legs to be shorted to each other. The buttons are arranged with B-C and A-D legs connected each to one (internally connected) vertical strip on the breadboard. Pressing the button will close the electrical connection between the two vertical strips. For the LED, the longer leg correspond to the positive terminal. See Fig. 8 below for illustrations.

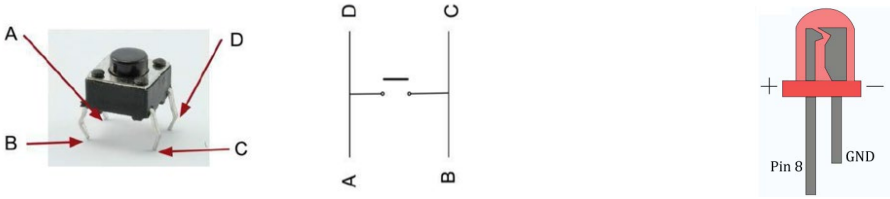


Fig. 8: Buttons, LED and connections

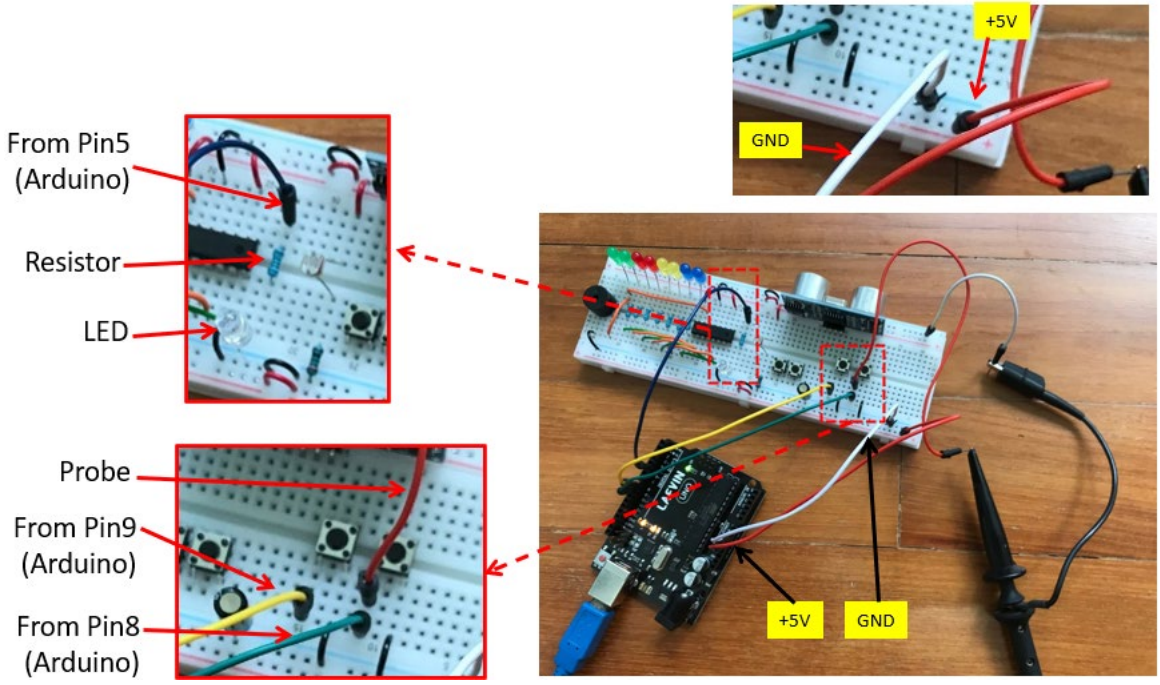


Fig 9: Hardware connections

6.3.2 Software

#Task: Open the sketch in C:\Cx1106\Labs\Lab3\Code\6.3\Digital_io. Compile and download the program to the Arduino. This code detects the button press and turning ON an LED.

#Question: What mode is the pin 'buttonApin' configured to in the following code?

```
int ledPin = 5;
int buttonApin = 9;
int buttonBpin = 8;

byte leds = 0;

void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(buttonApin, INPUT_PULLUP);
  pinMode(buttonBpin, INPUT_PULLUP);
}

void loop()
{
  if (digitalRead(buttonApin) == LOW)
  {
    digitalWrite(ledPin, HIGH);
  }
  if (digitalRead(buttonBpin) == LOW)
  {
    digitalWrite(ledPin, LOW);
  }
}
```

Fig 10: Sample code for button detection and LED ON/OFF.

The 'INPUT_PULLUP' mode setting will pull the input pin to a logic '1' by default. Hence, reading the pin status using digitalRead() will return a Logic '1'. When the button is pressed, the input pin will be shorted to ground and the digitalRead() will return a Logic '0'. The code will turn the LED ON when buttonA (pin 9) is pressed, LED will turn OFF when buttonB (Pin8) is pressed.

#Question:

- What is the voltage level at the processor pin 9 when digitalRead() returns a logic '0' and logic '1'?
- What is the voltage level at the processor pin 5 when digitalWrite() output a logic '0' and logic '1'?
- Verify your answer by measuring the DC voltage at pin 9 and the pin 5 of the processor. Section 6.3.3 below will walk you through the required steps.

6.3.3 Setting up an Oscilloscope to test DC voltage

Connect the oscilloscope probe as shown in Fig 11. The crocodile clip of the probe is connected to the ground of the breadboard and the hook of the probe is connected to the pin 9 of the processor. Note that we can probe from the breadboard since the button leg is electrically connected (shorted) to pin 9 of the processor on Arduino, similarly the ground. Follow the steps below to setup the scope for DC voltage measurements.

- Select/Enable Ch1 of the scope.
- Adjust the voltage scale to 5V/division.
- Set the trigger mode to 'Auto'.
- You should see the horizontal line appearing on the scope display. Centralise this to the middle of your scope using the vertical position knob.

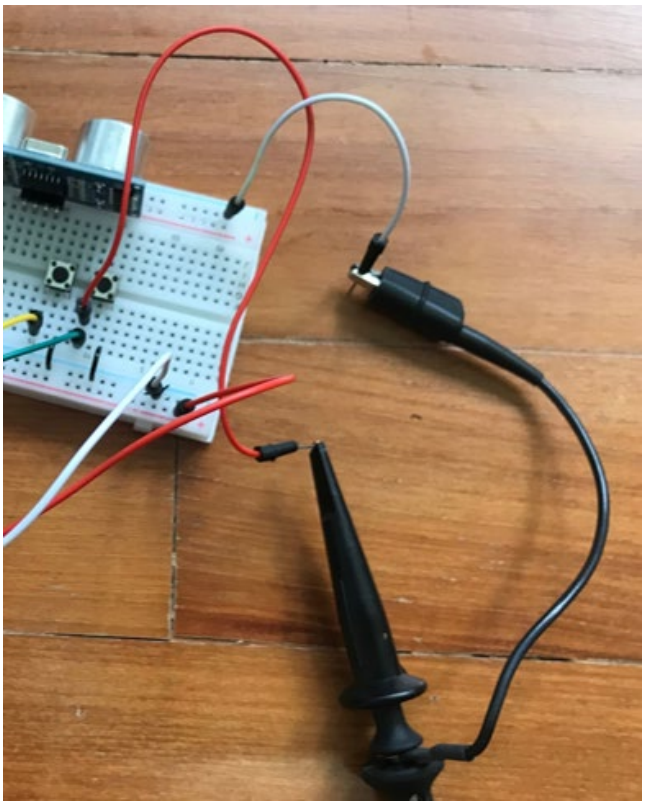


Fig.11: Probe connections

#Task: Proceed to measure the DC voltage of pin 5 when it output a logic '1'.

#Question: Where should you tap the pin 5 output on the breadboard?

6.4 Universal Asynchronous Receiver Transmitter (UART)

6.4.1 UART Overview

In this section, we will look at the UART interface in detail. This is an example of an asynchronous serial transfer interface that does not have a common clock line between receiver-transmitter so requires the synchronization information to be embedded in the data line. For UART case, the START, STOP and PARITY information is embedded together with the actual data payload in a UART frame.

To send data, e.g. an ASCII character via UART, the actual data payload may be configured to 7/8 data bits. The least significant bits are sent first. START and STOP bits are used to define the boundaries of the data packet. START is a single bit. STOP may be one or more bits, and depend on the configuration. Parity bit may be included to check the integrity of the transmitted data. Fig. 12 illustrates a sample waveform for upper-case letter J'.

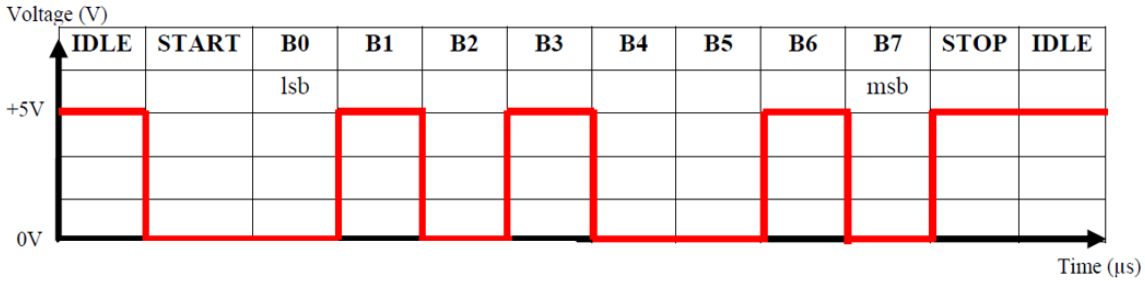


Fig.12: Sample Waveform for upper-case letter 'J'

The equivalent logic value of the transmitted data for letter 'J' is an 8-bit data with 0x4A. Comparing the data packet with the ASCII table shown in Table 2, it is seen 0x4A represents the upper-case letter 'J'. Hence, the data packet sent is the actual ASCII value for the character.

Note that the UART configuration used in Fig 12 is 8N1, i.e. 1 START, 8 DATA and 1 STOP bit. No parity scheme is used so the MSB of the Data is followed immediately by the STOP bit.

6.4.2 Hardware

#Task: Reuse the setup in section 6.3. Probe the UART TX pin of the processor. This is also pin 1 of the processor. You can access the processor pin 1 at the headers (See Fig 1). Fig 13 below shows the connection to Oscilloscope.

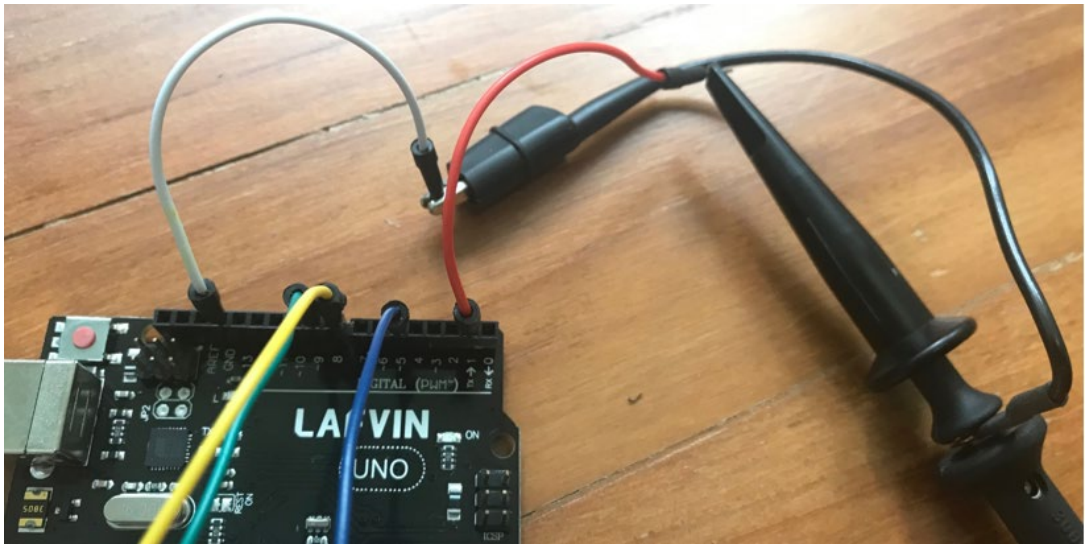


Fig. 13: Connecting Oscilloscope probe to Arduino to monitor UART TX

6.4.3 Software

#Task: Open the sketch in C:\Cx1106\Labs\Lab3\Code\6.4\uart_tx. Compile and download the program to the Arduino. Select ***Tools***>> ***Serial Monitor***. Ensure the serial monitor is configured to 19200 baud. The serial monitor should display the character 'U' every 1 second.

- Serial.begin() configure the UART port in the processor for transmission.
    - Serial.begin(19200, SERIAL_8E1) initialize the UART baud rate to 19200bps, 1 START, 8 DATA, Even Parity and 1 STOP bit.
- Serial.println() and Serial.write() prints to the UART port. This will appear in the Serial Monitor Tool of the Arduino Sketch Application on the PC.

```
int tx_char = 'U';

void setup() {
  // baud rate = 19200bps, 1 START, 8 DATA, Even Parity, 1 STOP bit
  Serial.begin(19200, SERIAL_8E1);
  while (!Serial);      // Wait until Serial is ready
  Serial.println("Serial Port ready");
}

void loop() {
  Serial.write(tx_char);//TX ASCII value of tx_char
  delay(1000);          //Delay for 1000ms
}
```

Fig. 14: UART Communication to send ASCII upper-case character 'U'

#Question:

- Where are the code created in the Arduino Sketch Application executed during runtime?
- Why do we need to configure the baud rate of the Serial Monitor in order for it to display the information correctly? What happen when the baud rate of the Serial Monitor is faster than that of the Arduino Processor UART?

6.4.4 Setting up an Oscilloscope to Probe UART TX signal

#Task: Now you need to display the UART TX signal on the scope.

- Ensure that connection shown in Fig. 13 is done.
- Set the voltage scale to 2V/divison.
- Set time base to 100µs/division
- Set to 'Normal' trigger mode.
- Set trigger source to ch1 (or whichever channel you use to probe the UART TX signal).
- Make any necessary vertical/horizontal positioning adjustment to bring the waveform to the center of the display.

#Task: Explore different UART configuration by modifying the input parameters of Serial.begin() function.

- Plot the waveform for the following configuration in Fig 15, 16 and 17. Identify the START bit, STOP bit(s) and the data payload. Also include the timing information, i.e. duration of one UART bit width.
    - Serial.begin(19200, **SERIAL_8E1**)

- Serial.begin(19200, **SERIAL_8O1**)
- Serial.begin(38400, **SERIAL_8E1**)

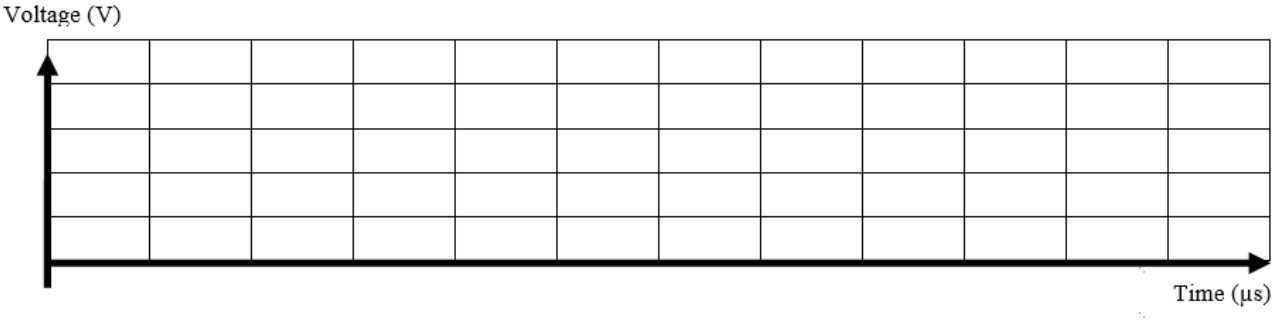


Fig. 15: Acquired Waveform for capitalized letter 'U'. Serial.begin(19200, **SERIAL_8E1**).

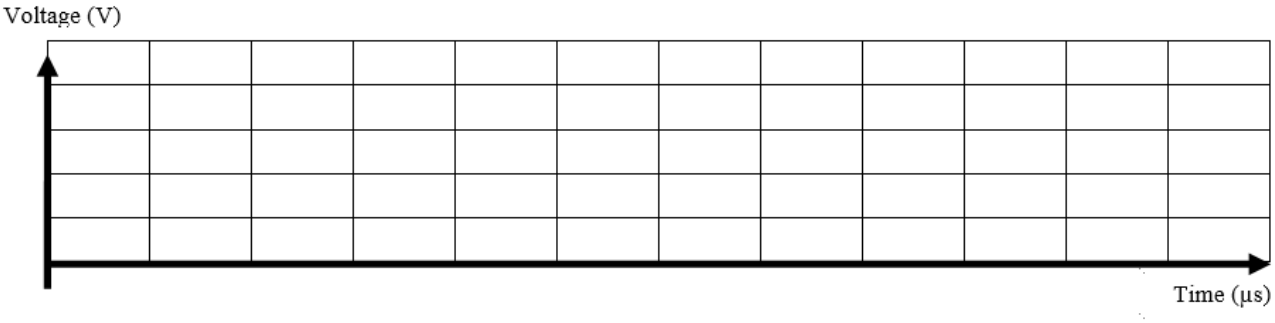


Fig. 16: Acquired Waveform for capitalized letter 'U'. Serial.begin(19200, **SERIAL_8O1**).

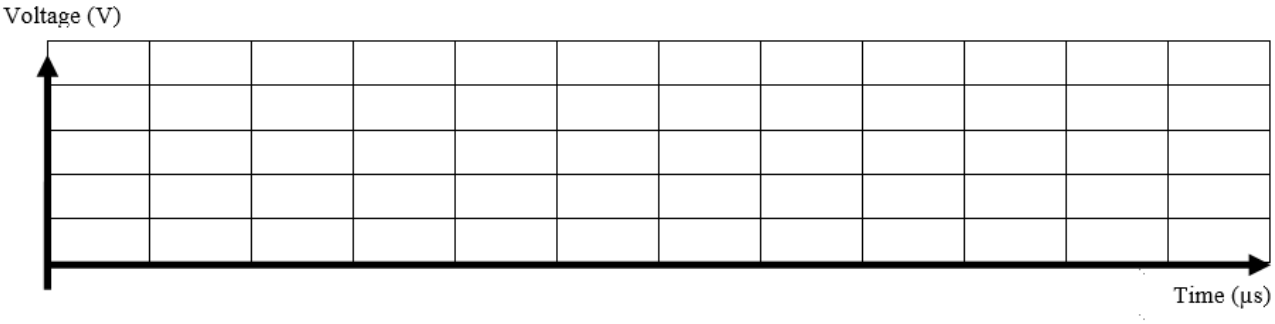


Fig. 17: Acquired Waveform for capitalized letter 'U'. Serial.begin(38400, **SERIAL_8E1**).

#Task: Note that it is also possible to initialize the UART with just "Serial.begin(19200)", i.e. not specifying the UART frame configuration. Try that and you will observe that the character 'U' is still received by the Serial Monitor and some waveform can be seen on the scope as well.

#Question: Given that the default data payload size is 8 bits. Find out what is the default parity scheme used (Even, Odd or None).

6.5 Synchronous Serial Data Transfer

In this section, you will look into the interface between the Arduino processor and a Serial-to-Parallel converter chip (74HC595). 74HC595 takes in a stream of serial data at its input and output a parallel equivalent after receiving 8 bits of serial data, see Fig. 18 below. This is an example of a proprietary synchronous serial data transfer. Receiver starts sampling the data line once it detected a rising edge at the clock line.

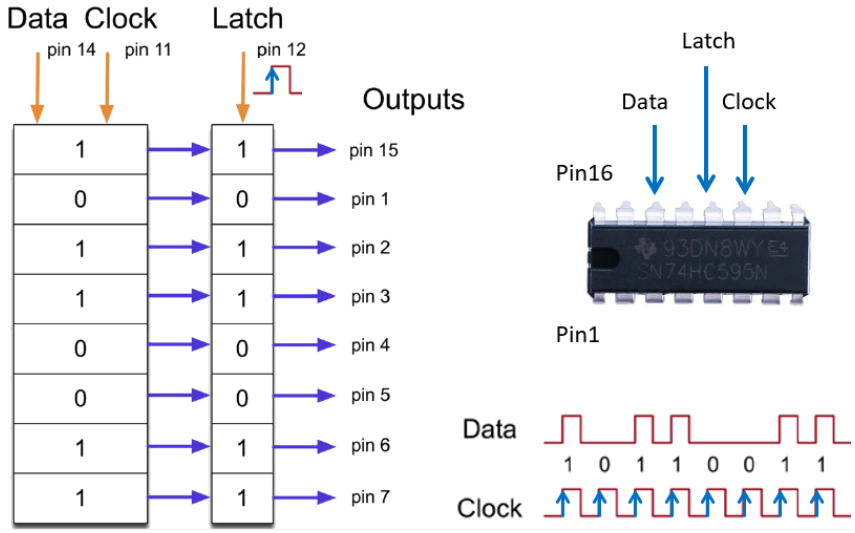


Fig. 18: Synchronous Serial Data Interface between processor and 74HC595

6.5.1 Hardware

#Task: Connect the circuit as per Fig 19. You only need to connect the Clock, Latch and Data line. The power, ground and other control signals are already connected beforehand. This is shown as the Yellow, Green and Dark Blue wires in the Photo, note that the graphical illustration uses a different colour scheme. Also, probe the Clock and Data signal as shown in the photo in Fig 19. Note: Pin9, 11 and 12 of Arduino should NEVER be connected directly to Ground or Vcc! The large current draw will damage the board and ICs.

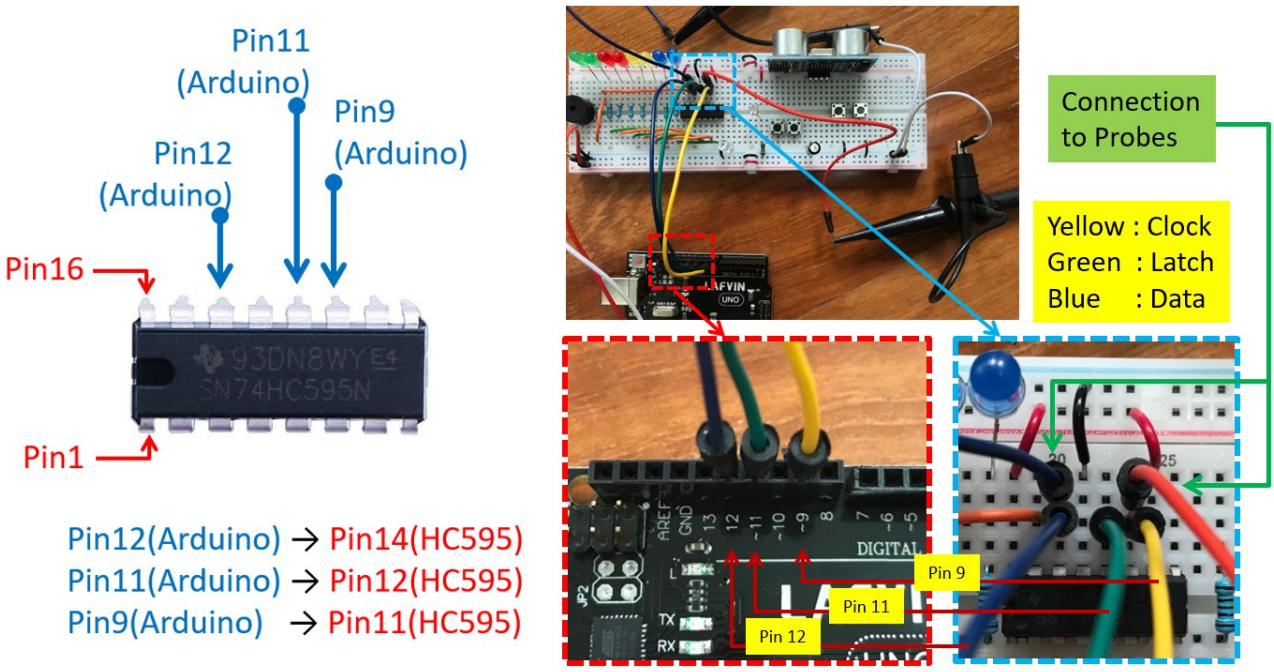


Fig. 19: Connecting Clock, Latch and Data signal of 74HC595 to Arduino Processor. Plus probe connections.

6.5.2 Software

#Task: Open the sketch in C:\Cx1106\Labs\Lab3\Code\6.5\SynchronousSerial. Compile and download the program to the Arduino.

#Question: Which bit of the 8-bit data is shifted out first? LSB or MSB?

```
int tDelay = 100;
int latchPin = 11;      // (11) ST_CP [RCK] on 74HC595
int clockPin = 9;       // (9) SH_CP [SCK] on 74HC595
int dataPin = 12;      // (12) DS [S1] on 74HC595

byte leds = 0;

void updateShiftRegister()
{
   digitalWrite(latchPin, LOW);
   shiftOut(dataPin, clockPin, LSBFIRST, leds);
   digitalWrite(latchPin, HIGH);
}

void setup()
{
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
}

void loop()
{
  leds = 0x5;
  updateShiftRegister();
  delay(tDelay);
}
```

Fig. 20: Proprietary Synchronous Serial Interface to 74HC595

6.5.3 Setting up an Oscilloscope to Probe Data and Clock TX signal

#Task: Now you need to display the Clock and Data signal on the scope.

- Ensure that connection shown in Fig. 19 is done.
- Enable Ch1 and Ch2 on scope. Connect Clock to Ch1 and Data to Ch2.
- Set the voltage scale to 5V/division.
- Set time base to 10µs/division
- Set to 'Normal' trigger mode.
- Set trigger source to ch1
- Make any necessary vertical/horizontal positioning adjustment to bring the waveform to the center of the display.

#Question: Why did we set the voltage scale to 5V/division? Can we set it to 1V/division instead?

#Task: Plot the Ch1 and Ch2 waveform obtained when transmitting a 0x5 value in Fig. 21 below.

- Indicate clearly when each bit of data is sampled by the receiver.

#Question:

- How does the receiver know when to sample the data from the data line? Indicate clearly when each bit of data is sampled by the receiver.
- What is the period of the clock signal?
- What is the maximum transfer rate achievable for this transmission?

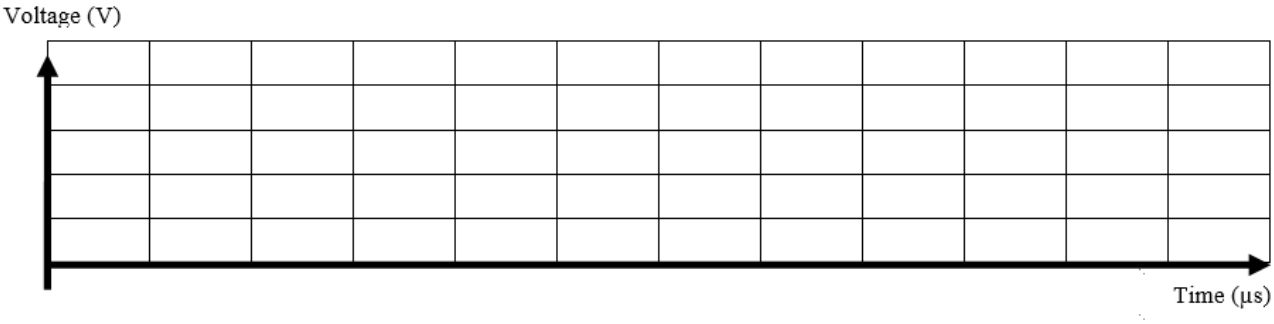


Fig. 21: Acquired Waveform (Clock and Data) when transmitting number 0x5.

#Task: Use the setup to find the negated value of 0x5, 0x7 and 0x9, i.e. -5, -7 and -9.

7. Optional Tasks Section

In this optional tasks section, you will test different input/output devices that can be used for different real world applications. Note again that this part will NOT be tested in the Lab quiz.

7.1 Tilt Ball Switch

Tilt sensors (tilt ball switch) allow you to detect orientation or inclination. They are small, inexpensive, low-power and easy-to-use. If used properly, they will not wear out. Their simplicity makes them popular for toys, gadgets and appliances. While not as precise or flexible as a full accelerometer, tilt switches can detect motion or orientation. Another benefit is that the big ones can switch power on their own. Accelerometers, on the other hand, output digital or analog voltage that must then be analyzed using extra circuitry.

#Task: Load the Arduino Sketch code in "7.1 Ball Switch" under "Optional Labs". Check the code and connect the appropriate pins to the Ball Switch and the White LED. If connection is correct, the while LED should turn ON/OFF when you tilt the breadboard.

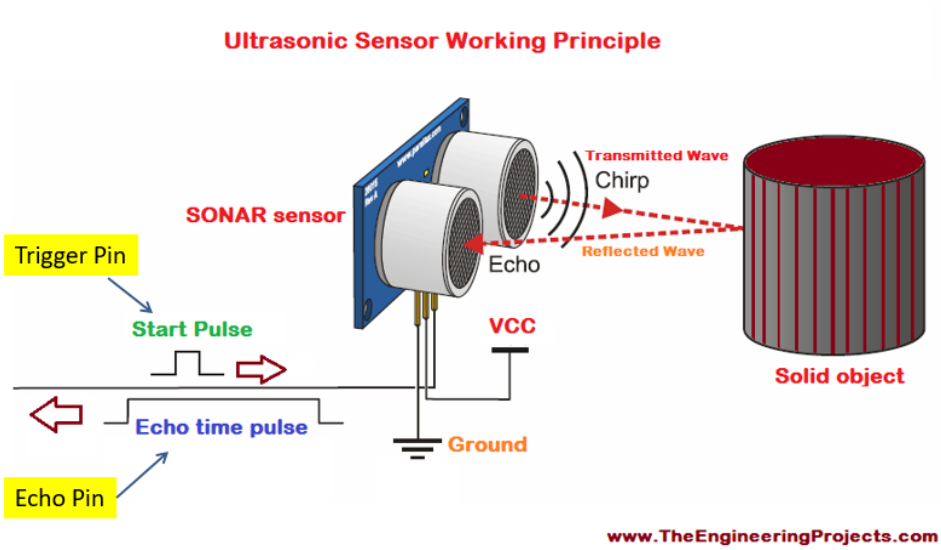
7.2 Passive Buzzer

The working principle of passive buzzer is using PWM generating audio to make the air to vibrate. Appropriately changed as long as the vibration frequency, it can generate different sounds. For example, sending a pulse of 523Hz, it can generate Alto Do, pulse of 587Hz, it can generate midrange Re, pulse of 659Hz, it can produce midrange Mi. Tuned correctly, you could get the buzzer to play a song!

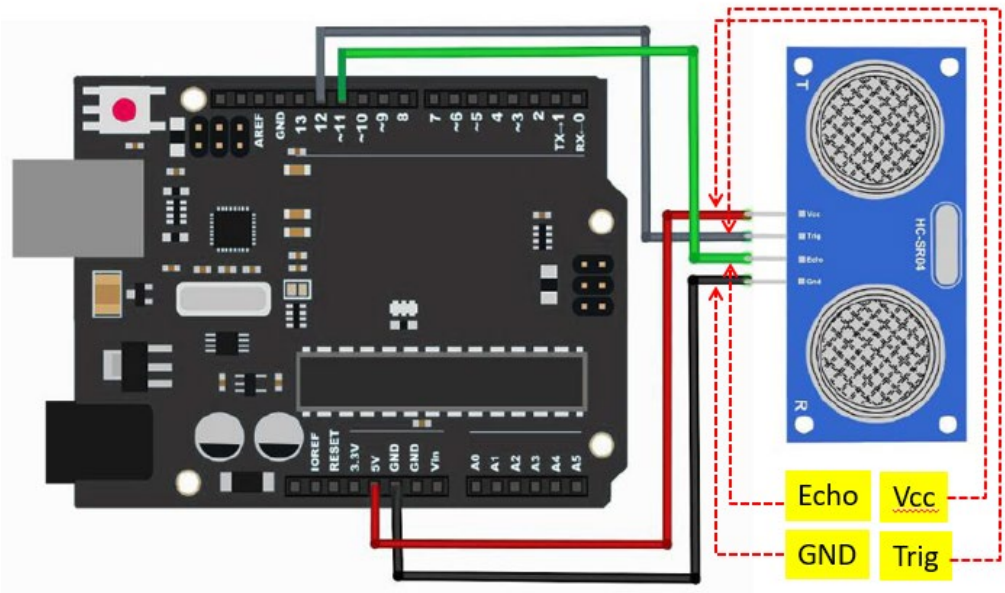
#Task: Load the Arduino Sketch code in "7.2 Passive Buzzer", connect the Positive Terminal of the Buzzer to Pin 8 of Arduino and create a melody.

7.3 Ultrasonic Sensor

Ultrasonic sensor module HC-SR04 provides 2cm-450cm non-contact measurement function. The sensor uses sonar to determine the distance to an object. To start the measurement, the trigger pin is set high for 10uS and then turned off. This will trigger an ultrasonic wave from the transmitter, once the wave is returned after it getting reflected by any object, the receiver will be able to pick it up. Internal circuitry will perform the necessary calculation and set the Echo pin high for a particular amount of time equal to the time taken for the wave to return back to the sensor. The time between the transmission and reception of the signal allows us to calculate the distance to an object as the velocity of sound in air is known.

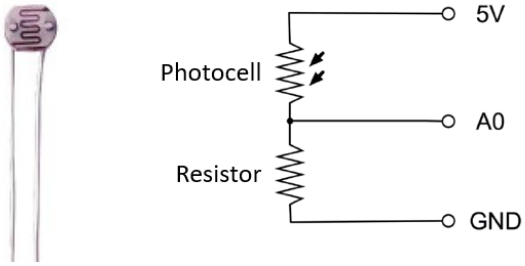


#Task: Connect the hardware as shown in the figure below. Load the Arduino Sketch code in "7.3 Ultrasonic", open the serial monitor and check if the distance reported is inline with the obstacles in front of the utlrasonic sensor. You can try to link it up with either the LEDs or Buzzer for display of distance or sound off an alarm when object is too close.



7.4 Photo Cell

The photocell used is of a type called a light dependent resistor. As the name suggests, these components act just like a resistor, except that the resistance changes in response to how much light is falling on them. This one has a resistance of about 50 kΩ in near darkness and 500 Ω in bright light. To convert this varying value of resistance into something we can measure on an UNO R3 board's analog input, it needs to be converted into a voltage. The simplest way to do that is to combine it with a fixed resistor.



#Task: Use the same hardware setup as section 6.5. Load the Arduino Sketch code in "7.4 Photocell", connect A0 (analog channel 0 pin) to the resistor divider point between photocell and the fixed valued resistor. See Figure below. Depending on the photocell characteristics and the fixed value resistor used, the voltage value read from A0 may differ. You can see the voltage level detect by opening the serial monitor. Gather the value when photocell is covered by your finger (dark) and not covered (bright). Adjust the divider ratio in the cod accordingly. If done correctly, the LED will show you the amount of light detected by the photocell.