Types of communication protocols
1) Sensor Interfacing. (Ultrasonic Sensor)
2) between two microcontrollers. (UART and I2C)
3) Ps2 controller interfacing
4) Get well versed in serial monitor Communication
5) Basic wireless communication protocols

SET-3

Figure 1. An Ultrasonic Sensor

By

AAYUSH KUMAR (Electrical)

1) SENSOR INTERFACING (ULTRASONIC SENSOR)

It emits an ultrasound at 40 000 Hz which travels through the air and if there is an object or obstacle on its path It will bounce back to the module. Considering the travel time and the speed of the sound you can calculate the distance.

The HC-SR04 Ultrasonic Module has 4 pins, Ground, VCC, Trig and Echo. The Ground and the VCC pins of the module needs to be connected to the Ground and the 5 volts pins on the Arduino Board respectively and the trig and echo pins to any Digital I/O pin on the Arduino Board.

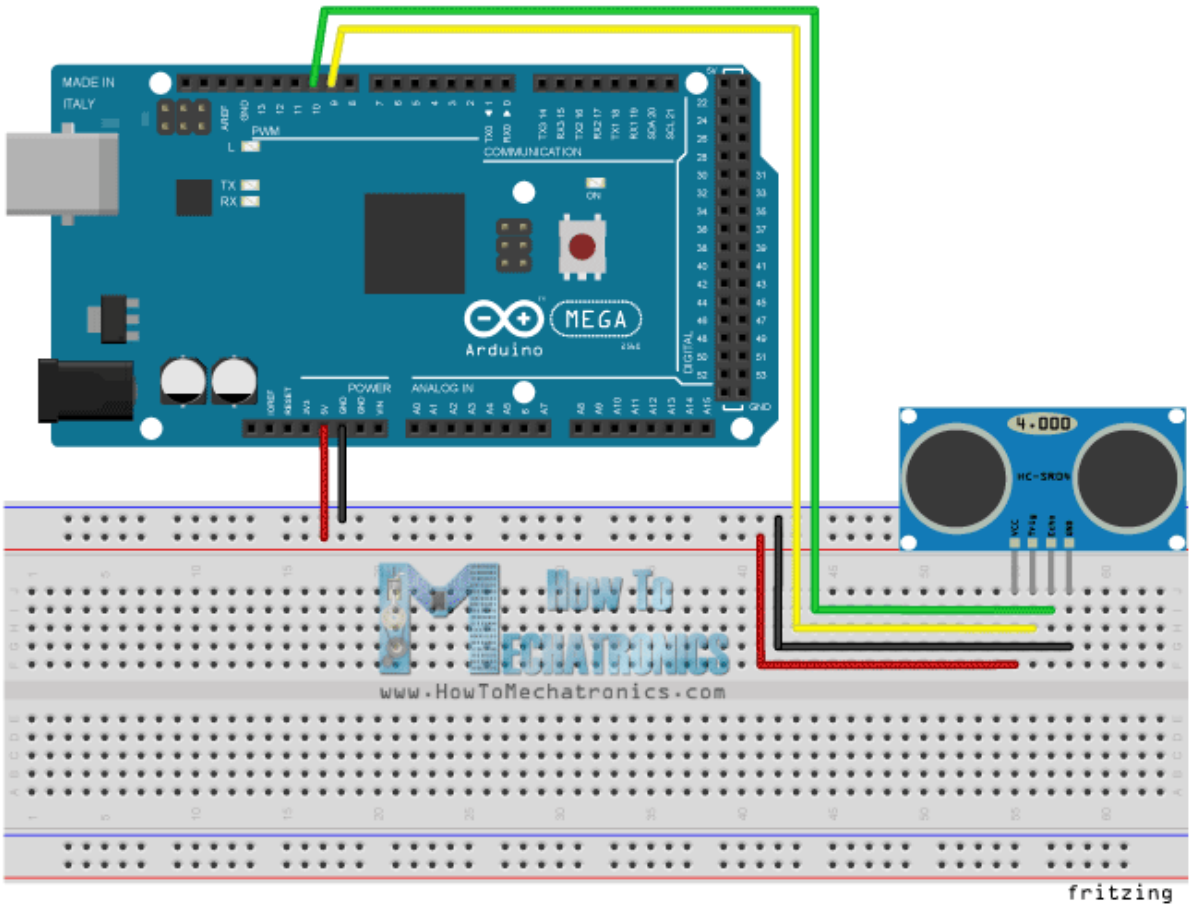

Figure 2. Wiring

In order to generate the ultrasound you need to set the Trig on a High State for 10 µs. That will send out an 8 cycle sonic burst which will travel at the speed sound and it will be received in the Echo pin. The Echo pin will output the time in microseconds the sound wave traveled.

For example, if the object is 10 cm away from the sensor, and the speed of the sound is 340 m/s or 0.034 cm/µs the sound wave will need to travel about 294 u seconds. But what you will get from the Echo pin will be double that number because the sound wave needs to travel forward and bounce backward. So in order to get the distance in cm we need to multiply the received travel time value from the echo pin by 0.034 and divide it by 2.

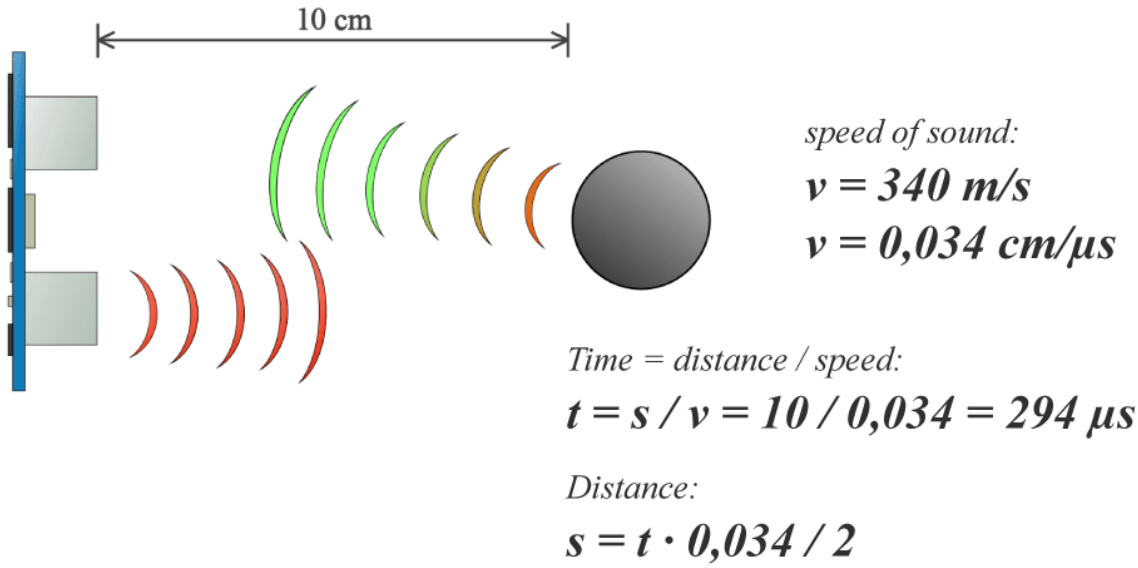

Figure 3. Calculating distance

The Code

```
// defines pins numbers
const int trigPin = 9;
const int echoPin = 10;

// defines variables
long duration;
int distance;

void setup() {
pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin, INPUT); // Sets the echoPin as an Input
```

```
12. Serial.begin(9600); // Starts the serial communication
13. }
14.
15. void loop() {
16. // Clears the trigPin
17. digitalWrite(trigPin, LOW);
18. delayMicroseconds(2);
19.
20. // Sets the trigPin on HIGH state for 10 micro seconds
21. digitalWrite(trigPin, HIGH);
22. delayMicroseconds(10);
23. digitalWrite(trigPin, LOW);
24.
25. // Reads the echoPin, returns the sound wave travel time in microseconds
26. duration = pulseIn(echoPin, HIGH);
27.
28. // Calculating the distance
29. distance= duration*0.034/2;
30.
31. // Prints the distance on the Serial Monitor
32. Serial.print("Distance: ");
33. Serial.println(distance);
34. }
```

2) BETWEEN TWO MICROCONTROLLERS (UART AND I2C)

UART

UART or Universal Asynchronous Receiver Transmitter is a dedicated hardware associated with serial communication. The hardware for UART can be a circuit integrated on the microcontroller or a dedicated IC. This is contrast to SPI or I2C, which are just communication protocols.

UART is one of the simplest and most commonly used Serial Communication techniques. Today, UART is being used in many applications like GPS Receivers, Bluetooth Modules, GSM and GPRS Modems, Wireless Communication Systems, RFID based applications etc.

BRIEF NOTE ON PARALLEL AND SERIAL COMMUNICATION

Transfer of Digital Data from one device to another can be achieved in two ways: Parallel Data Transfer and Serial Data Transfer. In parallel data transfer, all the bits are transferred from the source to destination at once.

This is possible because parallel data transfer uses multiple lanes or wires between the transmitter and receiver in order to transfer the data.

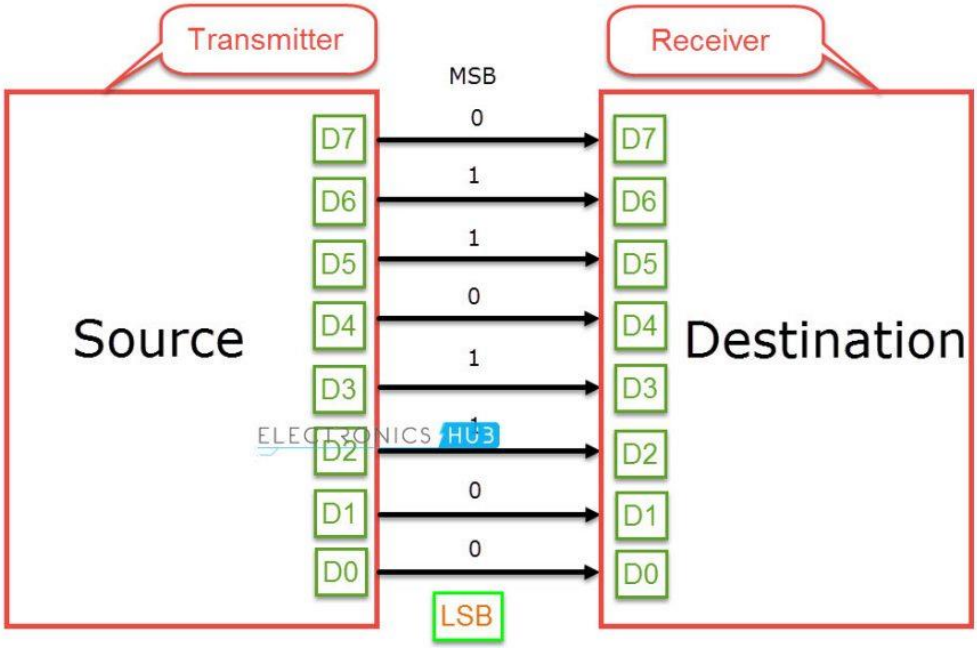

Figure 4. Parallel Communication

Parallel Data Transfer methods are faster and expensive as they needs more hardware and a lot of wires. Olden day's printers are the best example for external parallel communication. Other examples are RAM, PCI, etc.

With the progress in integrated circuit technology, the digital IC's are becoming smaller and faster and as a result the transfer rates in Parallel Communication with multiple lanes have reached a bottle neck.

Serial Communication on the hand, transfers data bit by bit using a single line or wire. For two way communication between the transmitter and receiver, we need just two wires for successful serial data transfer.

Since serial communication needs less circuitry and wires, the cost of implementing is less. As a result, using serial communication in complex circuitry might be more practical than parallel communication.

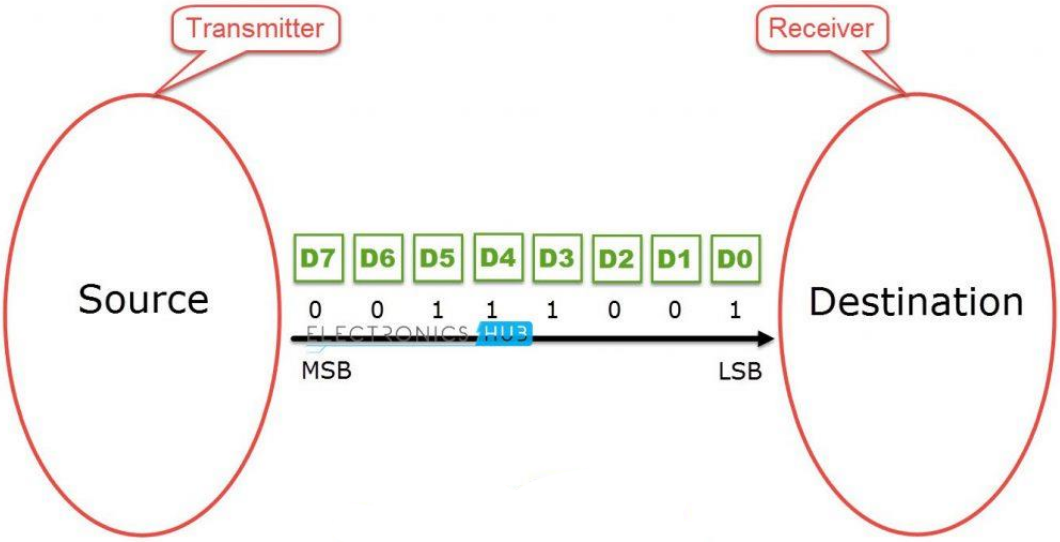

Figure 5. Serial Communication

But the only concern with serial data transfers is speed. Since the data transfer occurs over a single line, the speed of transfer in serial communication is less than that of parallel communication. Now – a – days, the speed of serial data transfer isn't a concern as advancements in technology have led to faster transfer speeds.

INTRODUCTION TO UART COMMUNICATION

UART or Universal Asynchronous Receiver Transmitter is a serial communication device that performs parallel – to – serial data conversion at the transmitter side and serial – to – parallel data conversion at the receiver side. It is universal because the parameters like transfer speed, data speed, etc. are configurable.

As mentioned in the introduction section, UART is a piece of hardware that acts as a bridge between the processor and the serial communication protocol

or port. The following image shows this interface briefly. The serial communication can be anything like USB, RS – 232, etc.

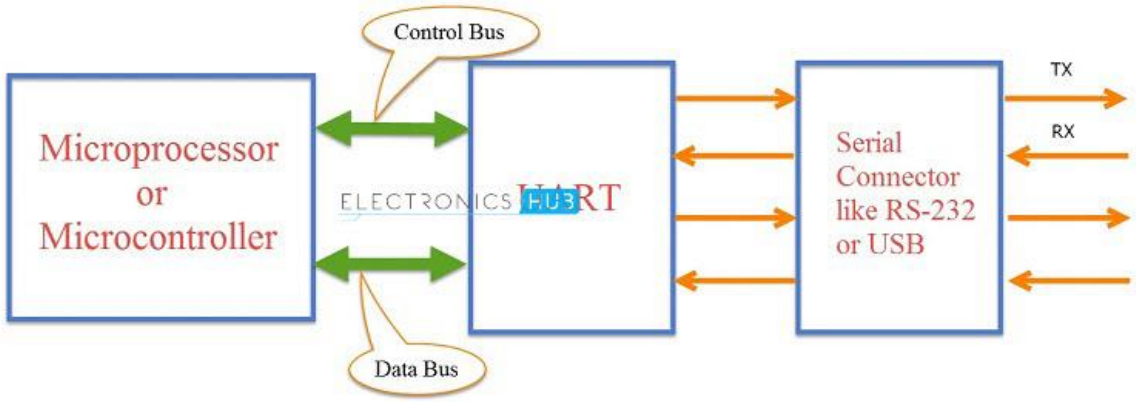

Figure 6. UART as a bridge

This is in contrast to Synchronous Serial Communication, which uses a clock signal that is shared between the transmitter and receiver in order to "Synchronize" the data between them.

If there is no clock (or any other timing signal) between the transmitter and receiver, then how does the receiver know when to read the data?

In UART, the transmitter and receiver must agree on timing parameters beforehand. Also, UART uses special bits at the beginning and ending of each data word to synchronize the transmitter and receiver. More about these special bits in the later sections.

In UART based Serial Communication, the transmitter and receiver communicate in the following manner. The UART on the sender device i.e. the transmitting UART receives parallel data from the CPU (microprocessor or microcontroller) and converts it in to serial data.

This serial data is transmitted to the UART on the receiver device i.e. receiving UART. The receiving UART, upon receiving the serial data, converts it back to parallel data and gives it to the CPU.

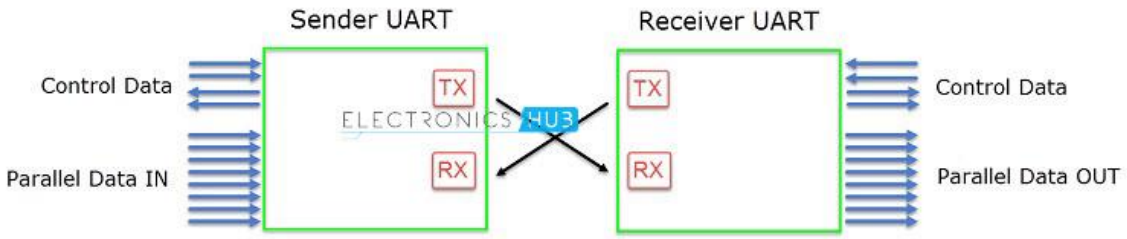

Figure 7. Communication

The pin on the transmitting UART, which transmits the serial data is called TX and the pin on the receiving UART, which receives the serial data is called RX.

Since the UART involves parallel – to – serial and serial – to – parallel data conversion, shift registers are an essential part of the UART hardware (two shift registers to be specific: Transmitter Shift Register and Receiver Shift Register).

WORKING

In UART Serial Communication, the data is transmitted asynchronously i.e. there is no clock or other timing signal involved between the sender and receiver. Instead of clock signal, UART uses some special bits called Start and Stop bits.

These bits are added to the actual data packet at the beginning and end respectively. These additional bits allows the receiving UART to identify the actual data.

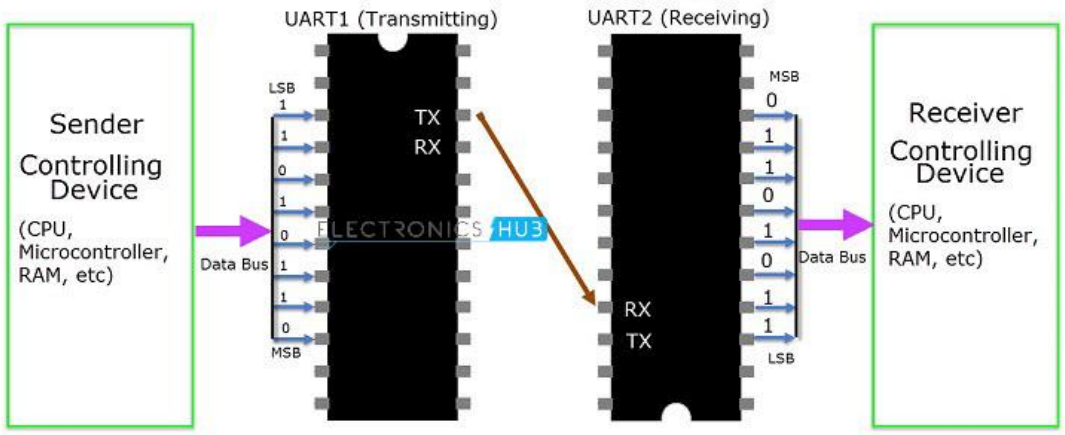

Figure 8. Working

The image above shows a typical UART connection. The transmitting UART receives data from the controlling device through the data bus. The controlling device can be anything like a CPU of a microprocessor or a microcontroller, memory unit like a RAM or ROM, etc. The data received by the transmitting UART from the data bus is parallel data.

To this data, the UART adds Start, Parity and Stop bits in order to convert it into a data packet. The data packet is then converted from parallel to serial with the help of shift register and is transmitted bit – by – bit from the TX pin.

The receiving UART receives this serial data at the RX pin and detects the actual data by identifying the start and stop bits. Parity bit is used to check the integrity of the data.

Up on separating the start, parity and stop bits from the data packet, the data is converted to parallel data with the help of shift register. This parallel data is sent to the controller at the receiving end through a data bus.

STRUCTURE OF DATA PACKET OR FRAME

The data in UART serial communication is organised in to blocks called Packets or Frames. The structure of a typical UART Data Packet or the standard framing of the data is shown in the following image.

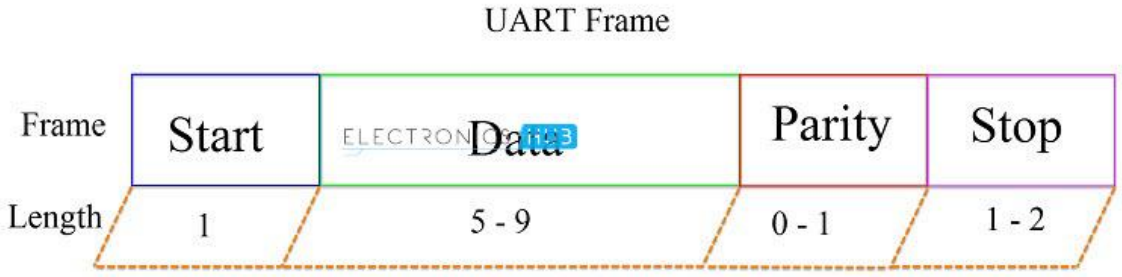

Figure 9. Structure of UART Data Packet

Let us see about each piece of the frame.

Start Bit: Start bit is a synchronisation bit that is added before the actual data. Start bit marks the beginning of the data packet. Usually, an idle data line i.e. when the data transmission line is not transmitting any data, it is held at a high voltage level (1).

In order to start the data transfer, the transmitting UART pulls the data line from high voltage level to low voltage level (from 1 to 0). The receiving UART detects this change from high to low on the data line and begins reading the actual data. Usually, there is only one start bit.

Stop Bit: The Stop Bit, as the name suggests, marks the end of the data packet. It is usually two bits long but often only on bit is used. In order to end the transmission, the UART maintains the data line at high voltage (1).

Parity Bit: Parity allows the receiver to check whether the received data is correct or not. Parity is a low – level error checking system and comes in two varieties: Even Parity and Odd Parity. Parity bit is optional and it is actually not that widely used.

Data Bits: Data bits are the actual data being transmitted from sender to receiver. The length of the data frame can be anywhere between 5 and 9 (9 bits if parity is not used and only 8 bits if parity is used). Usually, the LSB is the first bit of data to be transmitted (unless otherwise specified).

RULES OF UART

As mentioned earlier, there is no clock signal in UART and the transmitter and receiver must agree on some rules of serial communication for error free transfer of data. The rules include:

- Synchronization Bits (Start and Stop bits)
- Parity Bit
- Data Bits and
- Baud Rate

We have seen about synchronisation bits, parity bit and data bits. Another important parameter is the Baud Rate.
Baud Rate: The speed at which the data is transmitted is mentioned using Baud Rate. Both the transmitting UART and Receiving UART must agree on the Baud Rate for a successful data transmission.

Baud Rate is measured in bits per second. Some of the standard baud rates are 4800 bps, 9600 bps, 19200 bps, 115200 bps etc. Out of these 9600 bps baud rate is the most commonly used one.

Let us see an example data frame where two blocks of data i.e. 00101101 and 11010011 must be transmitted. The format of the frame is 9600 8N1 i.e. 9600 bps with 8 bits of data, no parity and 1 stop bit. In this example, we haven't used the parity bit.

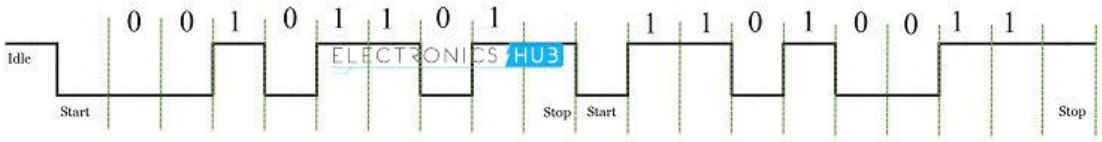

Figure 10. Example

ADVANTAGES OF UART

- Requires only two wires for full duplex data transmission (apart from the power lines).
- No need for clock or any other timing signal.

- Parity bit ensures basic error checking is integrated in to the data packet frame.

DISADVANTAGES OF UART

- Size of the data in the frame is limited.
- Speed for data transfer is less compared to parallel communication.
- Transmitter and receiver must agree to the rules of transmission and appropriate baud rate must be selected.

I2C

I2C communication is the short form for inter-integrated circuits. It is a communication protocol developed by Philips Semiconductors for the transfer of data between a central processor and multiple ICs on the same circuit board using just two common wires.

Owing to its simplicity, it is widely adopted for communication between microcontrollers and sensor arrays, displays, IoT devices, EEPROMs etc.

This is a type of synchronous serial communication protocol. It means that data bits are transferred one by one at regular intervals of time set by a reference clock line.

FEATURES

The following are some of the important features of I2C communication protocol:

- Only two common bus lines (wires) are required to control any device/IC on the I2C network
- No need of prior agreement on data transfer rate like in UART communication. So the data transfer speed can be adjusted whenever required
- Simple mechanism for validation of data transferred
- Uses 7-bit addressing system to target a specific device/IC on the I2C bus

- I2C networks are easy to scale. New devices can simply be connected to the two common I2C bus lines

HARDWARE

The physical I2C Bus

I2C Bus (Interface wires) consists of just two wires and are named as Serial Clock Line (SCL) and Serial Data Line (SDA). The data to be transferred is sent through the SDA wire and is synchronized with the clock signal from SCL. All the devices/ICs on the I2C network are connected to the same SCL and SDA lines as shown below:

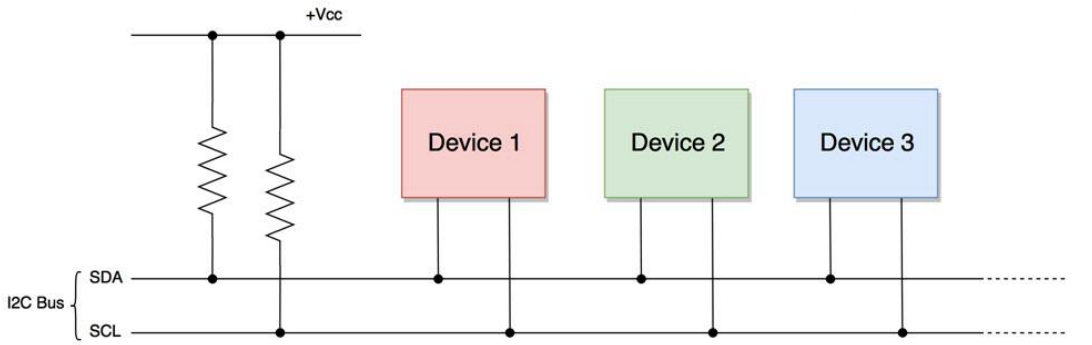

Figure 11. Connection

Both the I2C bus lines (SDA, SCL) are operated as open drain drivers. It means that any device/IC on the I2C network can drive SDA and SCL low, but they cannot drive them high. So, a pull up resistor is used for each bus line, to keep them high (at positive voltage) by default.

The reason for using an open-drain system is that there will be no chances of shorting, which might happen when one device tries to pull the line high and some other device tries to pull the line low.

Master and Slave Devices

The devices connected to the I2C bus are categorized as either masters or slaves. At any instant of time only a single master stays active on the I2C bus. It controls the SCL clock line and decides what operation is to be done on the SDA data line.

All the devices that respond to instructions from this master device are slaves. For differentiating between multiple slave devices connected to the same I2C bus, each slave device is physically assigned a permanent 7-bit address.

When a master device wants to transfer data to or from a slave device, it specifies this particular slave device address on the SDA line and then proceeds with the transfer. So effectively communication takes place between the master device and a particular slave device.

All the other slave devices doesn't respond unless their address is specified by the master device on the SDA line.

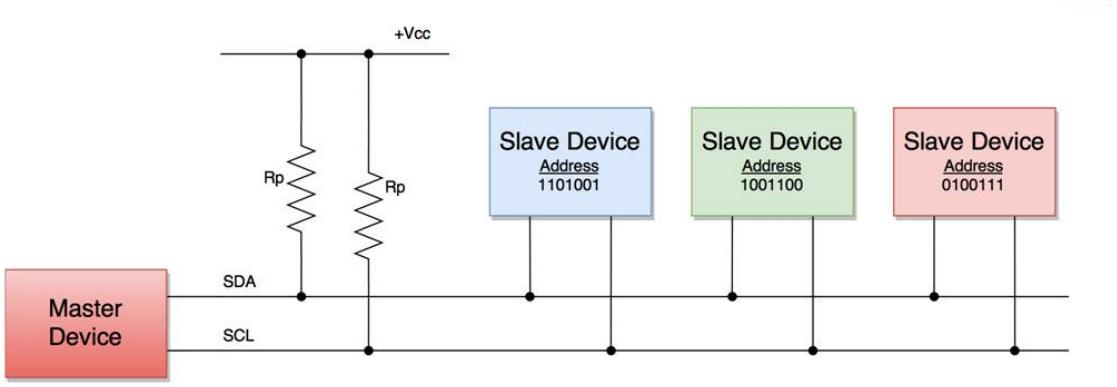

Figure 12. Communication

DATA TRANSFER PROTOCOL

The following protocol (set of rules) is followed by master device and slave devices for the transfer of data between them.

Data is transferred between the master device and slave devices through a single SDA data line, via patterned sequences of 0's and 1's (bits). Each sequence of 0's and 1's is termed as a transaction and the data in each transaction is structured as below:

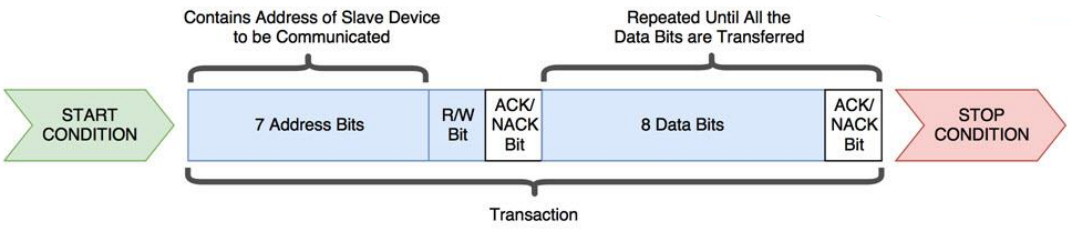

Figure 13. Transfer of Data

Start Condition

Whenever a master device/IC decides to start a transaction, it switches the SDA line from high voltage level to a low voltage level before the SCL line switches from high to low.

Once a start condition is sent by the master device, all the slave devices get active even if they are in sleep mode, and wait for the address bits.

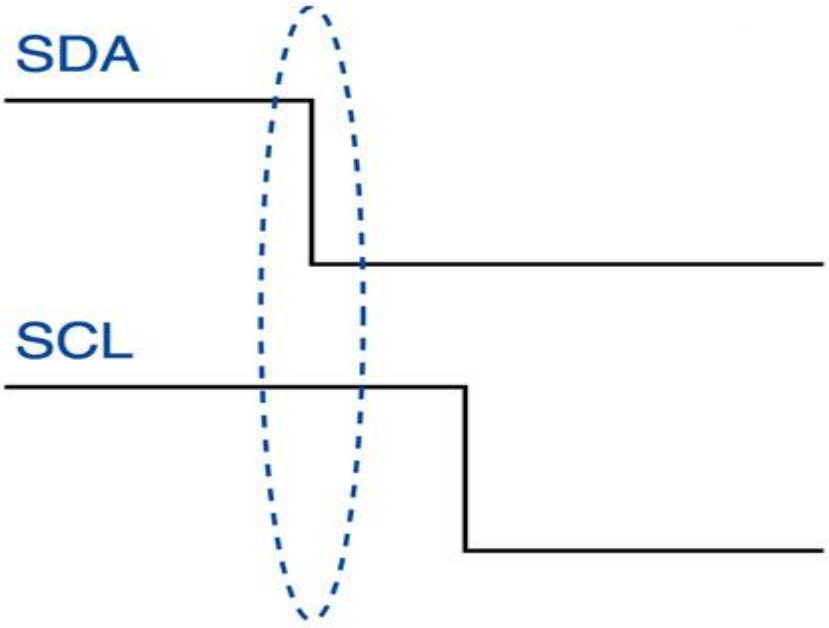

Figure 14. Start Condition

Address Block

It comprises of 7 bits and are filled with the address of slave device to/from which the master device needs send/receive data. All the slave devices on the I2C bus compare these address bits with their address.

Read/Write Bit

This bit specifies the direction of data transfer. If the master device/IC need to send data to a slave device, this bit is set to ‘0’. If the master IC needs to receive data from the slave device, it is set to ‘1’.

ACK/NACK Bit

It stands for Acknowledged/Not-Acknowledged bit. If the physical address of any slave device coincides with the address broadcasted by the master device, the value of this bit is set to ‘0’ by the slave device. Otherwise it remains at logic ‘1’ (default).

Data Block

It comprises of 8 bits and they are set by the sender, with the data bits it needs to transfer to the receiver. This block is followed by an ACK/NACK bit and is set to ‘0’ by the receiver if it successfully receives data. Otherwise it stays at logic ‘1’.

This combination of data block followed by ACK/NACK bit is repeated until the data is completely transferred.

Stop Condition

After required data blocks are transferred through the SDA line, the master device switches the SDA line from low voltage level to high voltage level before the SCL line switches from high to low.

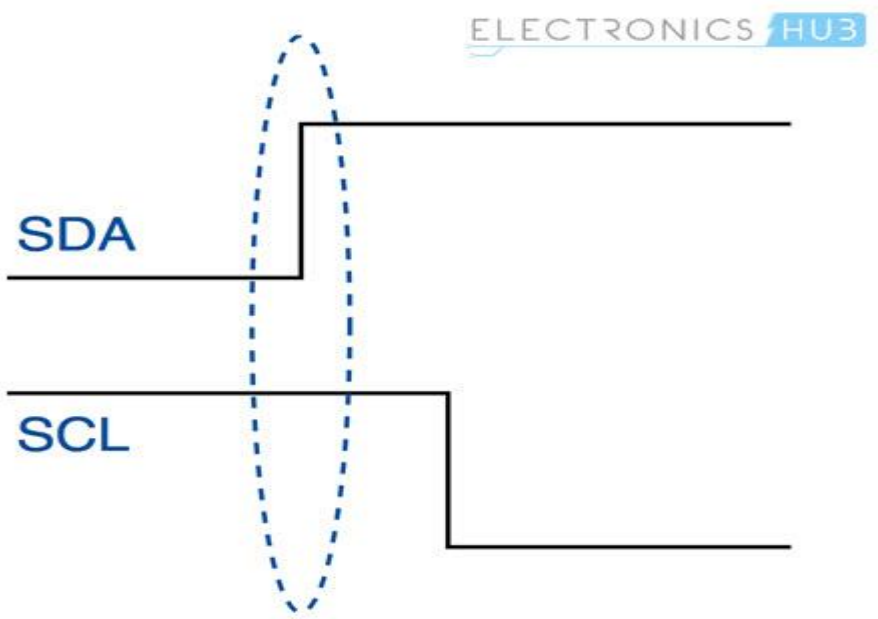

Figure 15. Stop Condition

NOTE: Logic '0' or setting a bit to '0' is equivalent to applying low voltage on the SDA line and vice versa.

WORKING

An I2C communication/transaction is initiated by a master device either to send data to a slave device or to receive data from it. Let us learn about working of both the scenarios in detail.

Sending Data to a Slave Device

The following sequence of operations take place when a master device tries to send data to a particular slave device through I2C bus:

- The master device sends the start condition
- The master device sends the 7 address bits which corresponds to the slave device to be targeted
- The master device sets the Read/Write bit to '0', which signifies a write
- Now two scenarios are possible:
    - If no slave device matches with the address sent by the master device, the next ACK/NACK bit stays at '1' (default). This

signals the master device that the slave device identification is unsuccessful. The master clock will end the current transaction by sending a Stop condition or a new Start condition

- If a slave device exists with the same address as the one specified by the master device, the slave device sets the ACK/NACK bit to '0', which signals the master device that a slave device is successfully targeted

- If a slave device is successfully targeted, the master device now sends 8 bits of data which is only considered and received by the targeted slave device. This data means nothing to the remaining slave devices
- If the data is successfully received by the slave device, it sets the ACK/NACK bit to '0', which signals the master device to continue
- The previous two steps are repeated until all the data is transferred
- After all the data is sent to the slave device, the master device sends the Stop condition which signals all the slave devices that the current transaction has ended

The below figure represents the overall data bits sent on the SDA line and the device that controls each of them:

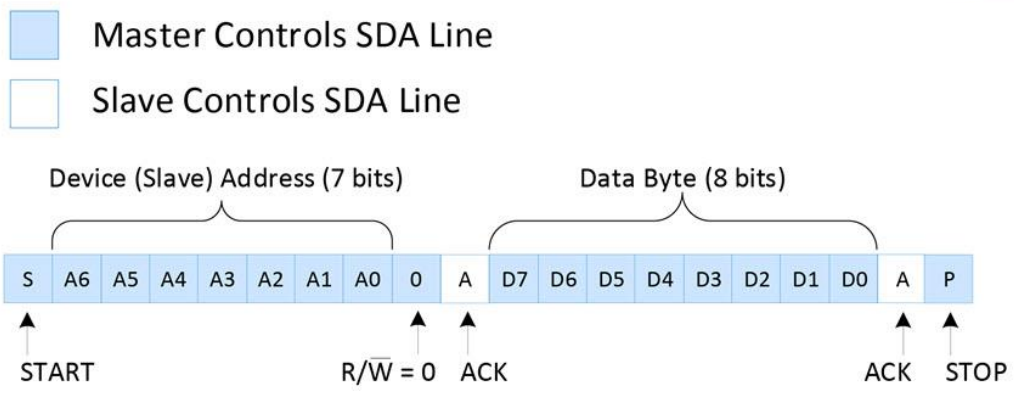

Figure 16. Data Bits

Reading Data from a Slave Device

The sequence of operations remain the same as in previous scenario except for the following:

- The master device sets the Read/Write bit to '1' instead of '0' which signals the targeted slave device that the master device is expecting data from it
- The 8 bits corresponding to the data block are sent by the slave device and the ACK/NACK bit is set by the master device
- Once the required data is received by the master device, it sends a NACK bit. Then the slave device stops sending data and releases the SDA line

If the master device to read data from specific internal l
device, it first sends the location data to the slave device u
previous scenario. It then starts the process of reading data with a repeated start condition.

Concept of clock stretching

Let say the master device started a transaction and sent address bits of a particular slave device followed by a Read bit of '1'. The specific slave device needs to send an ACK bit, immediately followed by data.

But if the slave device needs some time to fetch and send data to master device, during this gap, the master device will think that the slave device is sending some data.

To prevent this, the slave device holds the SCL clock line low until it is ready to transfer data bits. By doing this, the slave device signals the master device to wait for data bits until the clock line is released.

3) Ps2 controller interfacing

The PS2 wireless controller is a standard controller for the PlayStation 2 and is identical to the original DualShock controller for the PlayStation console. It features twelve analog (pressure-sensitive) buttons (X, O, Π, Δ, L1, R1, L2, R2, Up, Down, Left and Right), five digital button (L3, R3 Start, Select and the analog mode button) and two analog sticks. The controller also features two vibration motors, the left one being larger and more powerful than the one on the right. It is powered by two AAA batteries. It communicates with the console using 2.4 GHz RF protocol.

PS2 Receiver Pin Out

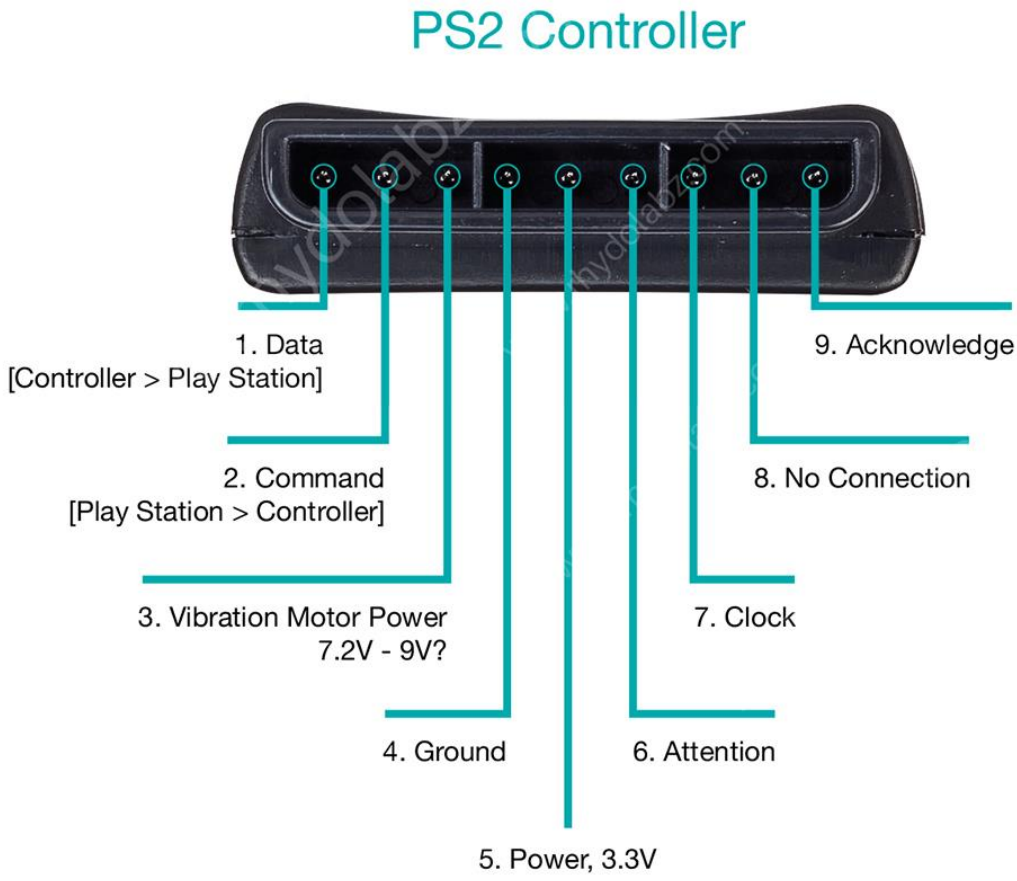

Figure 17. PS2 Controller

1. **DATA:** This is the data line from Controller to PS2. This is an open collector output and requires a pull-up resistor (1 to 10k, maybe more). (A pull-up resistor is needed because the controller can only connect this line to ground; it can't actually put voltage on the line).
2. **COMMAND:** This is the data line from PS2 to Controller.
3. **VIBRATION MOTOR POWER**
4. **GND:** Ground
5. **VCC:** VCC can vary from 5V down to 3V .

6. **ATT:** ATT is used to get the attention of the controller. This line must be pulled low before each group of bytes is sent / received, and then set high again afterwards. This pin consider as “Chip Select” or “Slave Select” line that is used to address different controllers on the same bus.
7. **CLK:** 500kH/z, normally high on. The communication appears to be SPI bus.
8. **Not Connected**
9. **ACK:** Acknowledge signal from Controller to PS2. This normally high line drops low about 12us after each byte for half a clock cycle, but not after the last bit in a set. This is a open collector output and requires a pull-up resistor (1 to 10k, maybe more).

PS2 Signals

PS2 wireless controller communicates with Arduino using a protocol that is basically SPI. The play station sends a byte at the same time as it receives one (full duplex) via serial communication. There’s a clock (SCK) to synchronize bits of data across two channels: DATA and CMD. Additionally, there’s a “Attention” (ATT) channel which tells the slave whether or not it is “active” and should listen to data bits coming across the CMD channel, or send data bits across the DATA channel (Reasonably, only one slave device should be active at a time) . The PlayStation 2 actually uses this plus an additional line that is not specifically part of the SPI protocol – an “Acknowledge” (ACK) line.

The clock is held high until a byte is to be sent. It then drops low (active low) to start 8 cycles during which data is simultaneously sent and received. The logic level on the data lines is changed by the transmitting device on the falling edge of clock. This is then read by the receiving device on the rising edge allowing time for the signal to settle. After each Command is received from the controller, that controller needs to pull ACK low for at least one clock cycle. If a selected controller does not ACK the PS2 will assume that there is no controller present. LSBs (least significant bits) are transmitting first.

Sample Project: Interfacing the PS2 Wireless Controller with Arduino

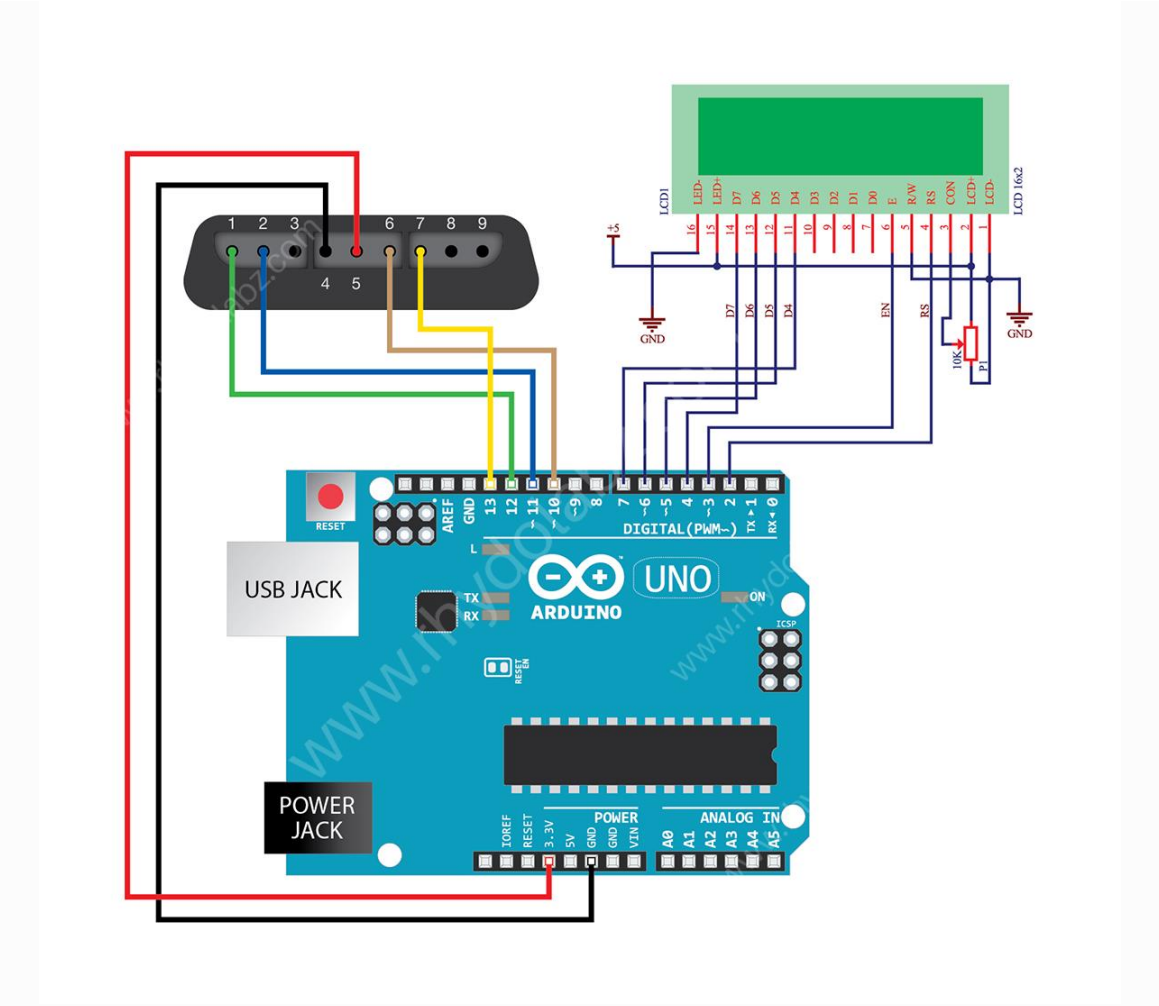

Figure 18. Circuit

Here we have interfaced the PS2 Wireless Controller with an Arduino. Upon each button press the Arduino receives the RF signal on the PS2 receiver and displays the it on the alphanumeric LCD module. We followed the standard PS2 protocol for realizing the communication algorithm, identical to the SPI protocol. Our program on the arduino detects and reads the button presses only, pressure values are not read. Analog stick state values are continuously displayed on the LCD module.

Connection Details

The PS2 receiver CLK line and ATT lines are held normally high. The ATT operates like the Slave Select line under SPI. You pull it low to tell the controller you are talking to it and then send it back high once a communications cycle is complete. CMD is the data line to the controller and DATA is the data coming from the controller. Here in our application we are not using the acknowledge pin.

Well that was so much until here, so here it is the photograph of what we actually did with that really awesome PS2 controller and the project code run on Arduino:

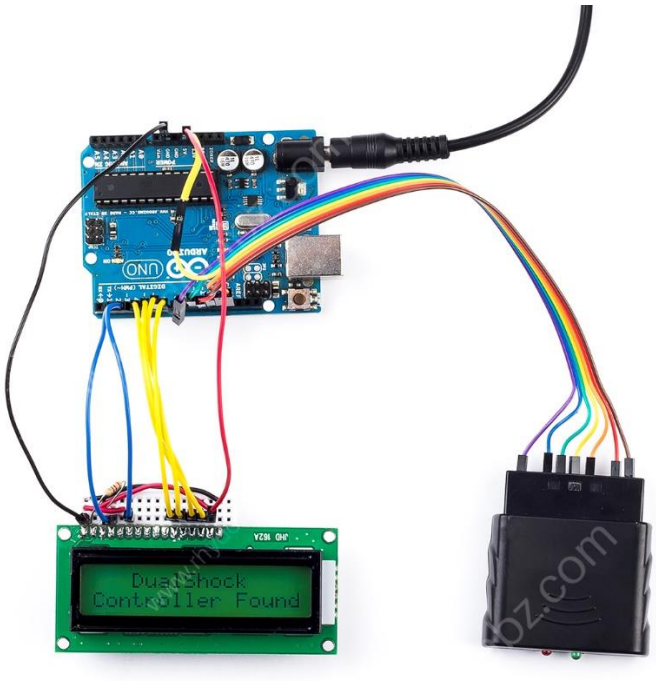

Figure 19. Wiring

Arduino Code

```
#include <PS2X_lib.h>                  /* PS2 Controller Library */
#include <LiquidCrystal.h>             /* LiquidCrystal Library */
PS2X ps2x;                             /* create PS2 Controller Class*/
byte Type = 0;
byte vibrate = 0;
```

```
int RX=0,RY=0,LX=0,LY=0;
LiquidCrystal lcd(2,3,7, 6, 5, 4);          /* initialize the library with the numbers of the interface pins*/
void setup(){
  lcd.begin(16, 2);                         /* 16X2 lcd display */
  ps2x.config_gamepad(13,11,10,12, true, true); /* setup pins and settings:  GamePad(clock, command, attention, data, Pressures?, Rumble?) check for error*/
  Type = ps2x.readType();                   /* Reading type of the PS2 Ccontroller */
  if(Type==1){                              /* Type 1 is Duel shock controller */
    lcd.setCursor(0, 0);                    /* Setting display position*/
    lcd.print("   DualShock    ");          /* display if the controller is duel shock*/
    lcd.setCursor(0, 1) ;
    lcd.print("Controller Found");
    delay(1000);
    lcd.clear();
  }
}
void loop(){

  ps2x.read_gamepad(false, vibrate);   /* read controller and set large motor to spin at 'vibrate' speed */
  lcd.setCursor(0, 0);                /* Position the LCD cursor */
  lcd.print("Stick values:   ");      /* Display analog stick values */
  lcd.setCursor(0, 1);
  LY = ps2x.Analog(PSS_LY);          /* Reading Left stick Y axis */
  LX = ps2x.Analog(PSS_LX);          /* Reading Left stick X axis */
  RY = ps2x.Analog(PSS_RY);          /* Reading Right stick Y axis */
  RX = ps2x.Analog(PSS_RX);          /* Reading Right stick X axis */
  if((LY <= 9))                     /* standardize to 3 digit by checking less than 10 */
    lcd.print("00");                /* eg: if LY= 5 then it display as "005" in lcd */
  if((LY >= 9 &&LY <= 99))          /* standardize to 3 digit by checking between 10-99 */
    lcd.print("0");                 /* eg: if LY= 55 then it display as "055" in lcd */
  lcd.print(LY,DEC);                /* display left analog stick Y axis */
  lcd.print(",");                   /* separate values using comma */
```

```
  if((LX <= 9))                 /* standardize to 3 digit by checking less than 10 */
    lcd.print("00");             /* eg: if LX= 5 then it display as "005" in lcd */
  if((LX >= 9 && LX<=99))         /* standardize to 3 digit by checking between 10-99 */
    lcd.print("0");              /* eg: if LX= 55 then it display as "055" in lcd */
  lcd.print(LX,DEC);              /* display left analog stick X axis */
  lcd.print(",");                /* separate values using comma */
  if((RY <= 9))                 /* standardize to 3 digit by checking less than 10 */
    lcd.print("00");             /* eg: if RY= 5 then it display as "005" in lcd */
  if((RY >= 9 &&RY<=99))          /* standardize to 3 digit by checking between 10-99 */
    lcd.print("0");              /* eg: if RY= 55 then it display as "055" in lcd */
  lcd.print(RY,DEC);              /* display Right analog stick Y axis             */
  lcd.print(",");                /* separate values using comma */
  if((RX <= 9))                 /* standardize to 3 digit by checking less than 10 */
    lcd.print("00");             /* eg: if RX= 5 then it display as "005" in lcd */
  if((RX >= 9 &&RX <= 99))        /* standardize to 3 digit by checking between 10-99 */
    lcd.print("0");              /* eg: if RX= 55 then it display as "055" in lcd */
  lcd.print(RX,DEC);              /* display Right analog stick X axis         */
  lcd.print(" ");
  if(ps2x.NewButtonState()) {       /* will be TRUE if any button changes state  */
    lcd.setCursor(0, 0);
    if(ps2x.Button(PSB_START))      /* will be TRUE as long START button is pressed */
      lcd.print("START PRESSED   ");
    if(ps2x.Button(PSB_SELECT))           /* will be TRUE as long SELECT button is
pressed */
      lcd.print("SELECT PRESSED  ");
    if(ps2x.Button(PSB_PAD_UP))           /* will be TRUE as long as UP button is
pressed */
      lcd.print("UP PRESSED      ");
    if(ps2x.Button(PSB_PAD_RIGHT))        /* will be TRUE as long as UP button is
pressed */
      lcd.print("RIGHT PRESSED   ");
    if(ps2x.Button(PSB_PAD_LEFT))         /* will be TRUE as long as LEFT button is
pressed */
      lcd.print("LEFT PRESSED    ");
```

```
    if(ps2x.Button(PSB_PAD_DOWN))          /* will be TRUE as long as DOWN button is
pressed */
      lcd.print("DOWN PRESSED    ");
    if(ps2x.Button(PSB_L1))               /* will be TRUE as long as L1 button is pressed */
      lcd.print("L1 pressed      ");
    if(ps2x.Button(PSB_R1))               /* will be TRUE as long as R1 button is pressed
*/
      lcd.print("R1 pressed      ");
    if(ps2x.Button(PSB_L2))               /* will be TRUE as long as L2 button is pressed */
      lcd.print("L2 pressed      ");
    if(ps2x.Button(PSB_R2))               /* will be TRUE as long as R2 button is pressed
*/
      lcd.print("R2 pressed      ");
    if(ps2x.Button(PSB_L3))               /* will be TRUE as long as L3 button is pressed */
      lcd.print("L3 pressed      ");
    if(ps2x.Button(PSB_R3))               /* will be TRUE as long as R3 button is pressed
*/
      lcd.print("R3 pressed      ");
    if(ps2x.Button(PSB_GREEN))            /* will be TRUE as long as GREEN/Triangle
button is pressed */
      lcd.print("Triangle pressed");
    if(ps2x.Button(PSB_BLUE))             /* will be TRUE as long as BLUE/CROSS/X
button is pressed */
      lcd.print("X pressed       ");
     if(ps2x.Button(PSB_RED))             /* will be TRUE as long as RED/Circle button is
pressed */
      lcd.print("Circle pressed  ");
    if(ps2x.Button(PSB_PINK))             /* will be TRUE as long as PINK/Squre button is
pressed */
      lcd.print("Square pressed  ");
    delay(700);
  }
  else;
}
```

This is the link to the PS2X library which is used in the above code:

https://codeload.github.com/madsci1016/Arduino-PS2X/legacy.zip/5d2be701af64d826d268301d83119a6d2ad04f15

4) SERIAL MONITOR COMMUNICATION

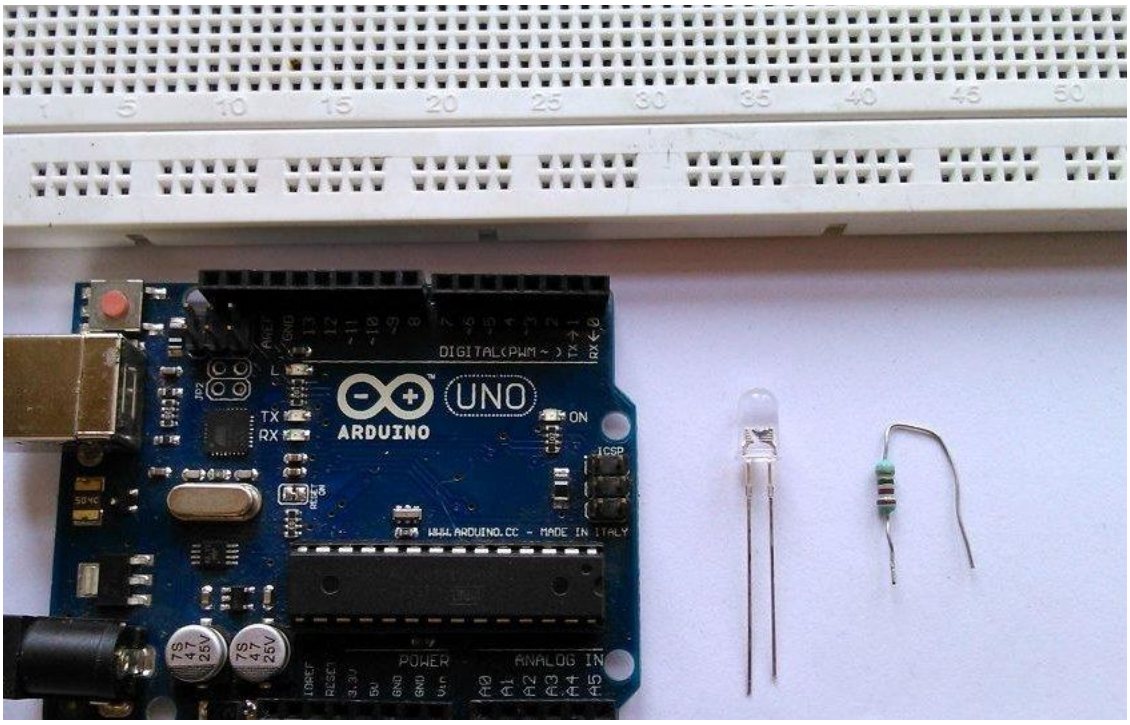

Figure 20. Components

Requirements

- A breadboard
- A LED

- A 220 ohm resistor
- An Arduino

The Code

```
void setup()

{

pinMode(13, OUTPUT);

Serial.begin(9600);

while (!Serial);

Serial.println("Input 1 to Turn LED on and 2 to off");

}

void loop() {

if (Serial.available())

{

int state = Serial.parseInt();

if (state == 1)

{

digitalWrite(13, HIGH);

Serial.println("Command completed LED turned ON");

}

if (state == 2)
```

```
{
digitalWrite(13, LOW);
Serial.println("Command completed LED turned OFF");
}
}
}
```

Open Serial Monitor, sending 1 turns on the LED and sending 2 turns off the LED.

5) BASIC WIRELESS COMMUNICATION PROTOCOLS

In the present days, wireless communication system has become an essential part of various types of wireless communication devices, that permits user to communicate even from remote operated areas. There are many devices used for wireless communication like mobiles. Cordless telephones, Zigbee wirelss technology, GPS, Wi-Fi, satellite television and wireless computer parts. Current wireless phones include 3 and 4G networks, Bluetooth and Wi-Fi technologies.
The different types of wireless communication mainly include, IR wireless communication, satellite communication, broadcast radio, Microwave radio, Bluetooth, Zigbee etc.

Satellite Communication

Satellite communication is one type of self contained wireless communication technology, it is widely spread all over the world to allow users to stay

connected almost anywhere on the earth. When the signal (a beam of modulated microwave) is sent near the satellite then, satellite amplifies the signal and sent it back to the antenna receiver which is located on the surface of the earth. Satellite communication contains two main components like the space segment and the ground segment.The ground segment consists of fixed or mobile transmission, reception and ancillary equipment and the space segment, which mainly is the satellite itself.

Figure 21. Satellite Communication

Infrared Communication

Infrared wireless communication communicates information in a device or systems through IR radiation . IR is electromagnetic energy at a wavelength that is longer than that of red light. It is used for security control, TV remote control and short range communications. In the electromagnetic spectrum, IR radiation lies between microwaves and visible light. So, they can be used as a source of communication

Figure 22. Infrared Communication

For a successful infrared communication, a photo LED transmitter and a photo diode receptor are required. The LED transmitter transmits the IR signal in the form of non visible light, that is captured and saved by the photoreceptor. So the information between the source and the target is transferred in this way. The source and destination can be mobile phones, TVs, security systems, laptops etc supports wireless communication.

Broadcast Radio

The first wireless communication technology is the open radio communication to seek out widespread use, and it still serves a purpose nowadays. Handy multichannel radios permit a user to speak over short distances, whereas citizen's band and maritime radios offer communication services for sailors. Ham radio enthusiasts share data and function emergency communication aids throughout disasters with their powerful broadcasting gear, and can even communicate digital information over the radio frequency spectrum.

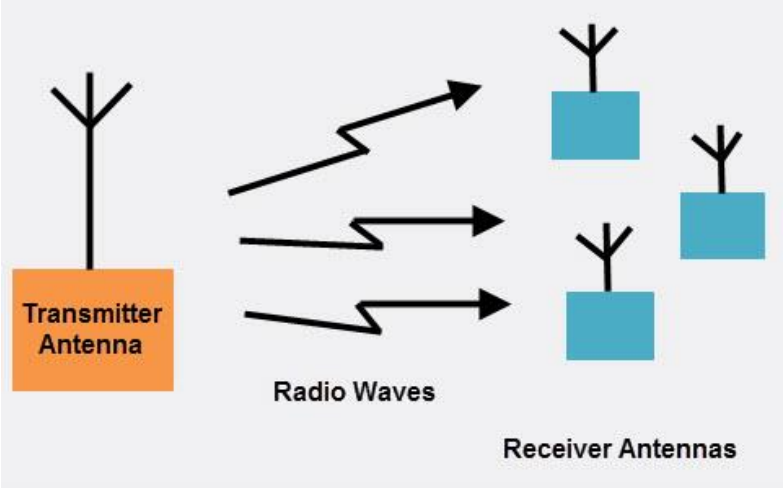

Figure 23. Broadcast Radio

Mostly an audio broadcasting service, radio broadcasts sound through the air as radio waves. Radio uses a transmitter which is used to transmit the data in the form of radio waves to a receiving antenna(Different Types of Antennas). To broadcast common programming, stations are associated with the radio N/W's. The broadcast happens either in simulcast or syndication or both. Radio broadcasting may be done via cable FM, the net and satellites. A broadcast sends information over long distances at up to two megabits/Sec (AM/FM Radio).
Radio waves are electromagnetic signals, that are transmitted by an antenna.These waves have completely different frequency segments, and you will be ready to obtain an audio signal by changing into a frequency segment.

For example, you can take a radio station. When the RJ says you are listening to 92.7 BIG FM, what he really means is that signals are being broadcasted at a frequency of 92.7megahertz, that successively means the transmitter at the station is periodic at a frequency of 92.700,000 Cycles/second.

When you would like to listen to 92.7 BIG FM, all you have to do is tune the radio to just accept that specific frequency and you will receive perfect audio reception.

Microwave Communication

Microwave wireless communication is an effective type of communication, mainly this transmission uses radio waves, and the wavelengths of radio waves are measured in centimeters. In this communication, the data or information can be transfers using two methods. One is satellite method and another one is terrestrial method.

Figure 24. Microwave Communication

Wherein satellite method, the data can be transmitted though a satellite, that orbit 22,300 miles above the earth. Stations on the earth send and receive data signals from the satellite with a frequency ranging from 11GHz-14GHz and with a transmission speed of 1Mbps to 10Mbps. In terrestrial method, in which two microwave towers with a clear line of sight between them are used, ensuring no obstacles to disrupt the line of sight. So it is used often for the purpose of privacy. The frequency range of the terrestrial system is typically 4GHz-6GHz and with a transmission speed is usually 1Mbps to 10Mbps.

The main disadvantage of microwave signals is, they can be affected by bad weather, especially rain.

Wi-Fi

Wi-Fi is a low power wireless communication, that is used by various electronic devices like smart phones, laptops, etc.In this setup, a router works as a communication hub wirelessly. These networks allow users to connect only within close proximity to a router. WiFi is very common in networking applications which affords portability wirelessly. These networks need to be protected with passwords for the purpose of security, otherwise it will access by others

Mobile Communication Systems

The advancement of mobile networks is enumerated by generations. Many users communicate across a single frequency band through mobile phones. Cellular and cordless phones are two examples of devices which make use of wireless signals. Typically, cell phones have a larger range of networks to provide a coverage.But, Cordless phones have a limited range. Similar to GPS devices, some phones make use of signals from satellites to communicate.

Figure 25. Mobile Communication

Bluetooth Technology

The main function of the Bluetooth technology is that permits you to connect a various electronic devices wirelessly to a system for the transferring of data.Cell phones are connected to hands free earphones, mouse, wireless keyboard. By using Bluetooth device the information from one device to another device. This technology has various functions and it is used commonly in the wireless communication market.

Figure 26. Bluetooth Technology

Advantages of Wireless Communication

- Any data or information can be transmitted faster and with a high speed
- Maintenance and installation is less cost for these networks.
- The internet can be accessed from anywhere wirelessly
- It is very helpful for workers, doctors working in remote areas as they can be in touch with medical centers.

Disadvantages of Wireless Communication

- An unauthorized person can easily capture the wireless signals which spread through the air.
- It is very important to secure the wireless network so that the information cannot be misused by unauthorized users

Applications of Wireless Communication

Applications of wireless communication involve security systems, television remote control, Wi-Fi, Cell phones, wireless power transfer, computer interface devices and various wireless communication based projects.

- Bluetooth
- WiFi
- Zigbee
- Low Cost 802.15.4
- GPRS / LTE
- RFID / Near Field Communications
- GPS
- 6loWPAN (802.15.4 radio with IPv6 packets and addressing)
- LoRa

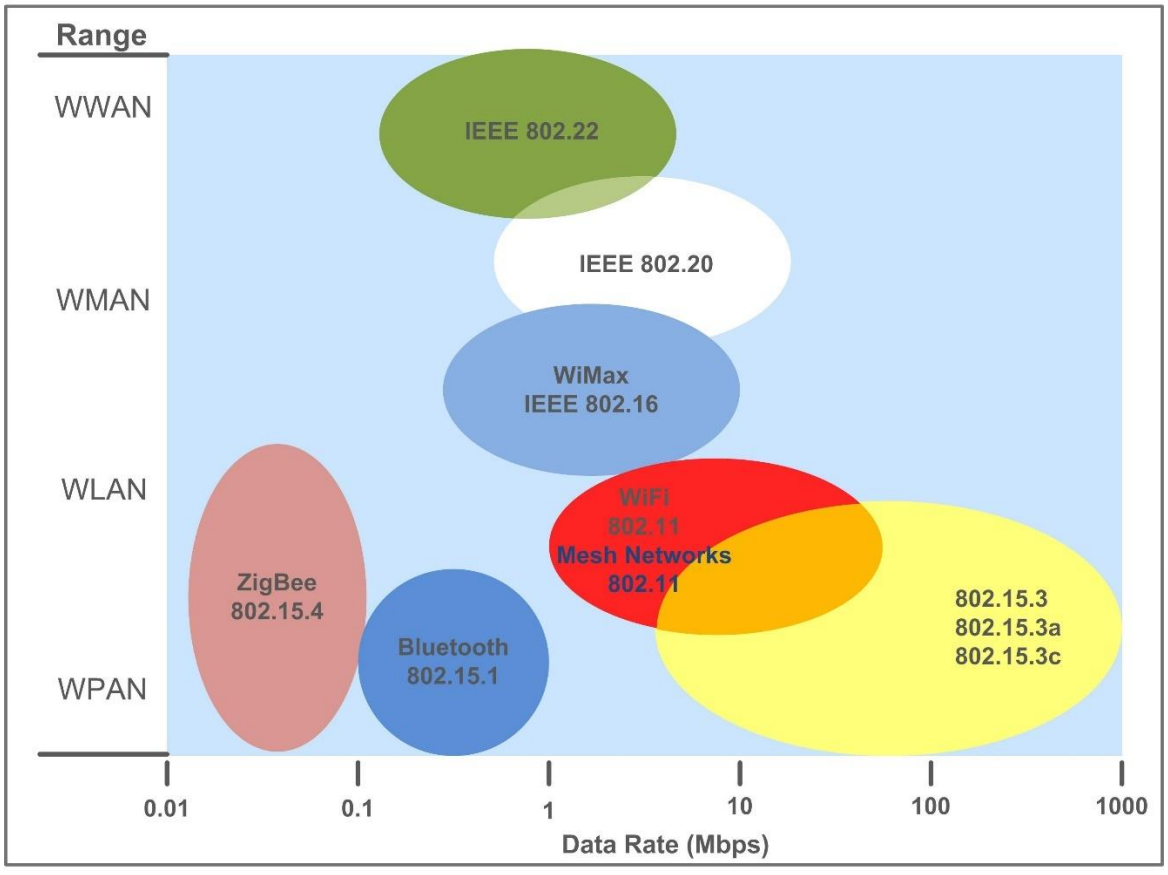

Figure 27. Data Rate

The RoweBots Bluetooth stack comes in two flavours: an AT command set or spi interface module and an HCI interface module. The advantage of the AT or SPI interface is that the entire protocol stack can run on the wireless module. Typically this is done for SPP or serial profiles. In contrast the HCI interface approach runs the balance of the protocol stack on the MCU or MPU and then the developer has a much broader set of choices for the profile that is used, including audio, health data, handsfree and much more. These wireless protocols offer complete end to end protocol solutions with full interoperability and POSIX device interfaces.

WiFi wireless protocols provide local area networking using a TCP/IP protocol stack with either IPV4 or IPV6. As a result, WiFi support is transformed into a network driver for these stacks with extra interfaces to choose a wireless network or terminate a wireless network and select another network for communication.

Zigbee and 802.15.4 low cost wireless protocols are very similar. The radio module used is identical and the zigbee stack is provided by the module developer along with the various low cost wireless options. These protocols are ideally suitable for a UDP or data gram sockets based interface. This plays to the strength of these wireless networks which are subject to interference and signal dropout. Demonstration programs provide seamless integration and operation in 10 minutes out of the box.

GPRS and LTE radios are supported in modules accessing the cell network to provide very wide area networking. Typically these connections are setup using an AT command set or SPI interface to establish a call and then TCP/IP and PPP are used to provide a full networking service over the GPRS radio.

Support for GPS and RFID / Near Field Communications wireless protocols using standard radio modules are easily intergrated into the system. Using standard modules interfaces range from SPI to AT command sets. Standard POSIX file interfaces are provided to read data from these devices.

6loWPAN isa new wireless protocol. It provides TCP and UDP services over 802.15.4 mesh radio networks. It integrates ideas associated with low power and intermittent node up times to minimize power consumption using two other new protocols called ROLL and RPL. Interesting the idea of routing at the mesh level has been rejected in favour of dynamic routing at the IP level.

Contact us for further information on our wireless protocols and our partner's modules.

Bluetooth v4

Bluetooth v4 is the latest version of bluetooth and it is used in a wide variety of applications. It has essentially the same structure and services as previous bluetooth versions with the exception of support for low power sensor applications with fast startup.

Unison Bluetooth v2.1

Unison Bluetooth has support for Bluetooth version 2.1 with a comprehensive set of profiles. It is fast and easy to use and configure for specific applications. It comes with assymetric profile support as illustrated in the video above. For example, both ends of the headset profile are supported so headset devices may be supported or devices which require headsets may be supported.

Unison Bluetooth can be setup to work in three ways:

- It can run inside a module and provide bluetooth support to users typically accepting commands through an AT command set.. The module developer has to implement this solution and it is not intended for anyone but module developers.
- Unison Bluetooth can provide the high level support on an MPU, MCU or FPGA to use a module with a complete Bluetooth stack on board.
- Unison Bluetooth can run on the MCU, MPU or FPGA and provide the complete stack from the HCI interface upwards. This is the most common way to run Unison Bluetooth because it can then have any profile that the user wants or even combined profiles.

Typically modules put SPP or serial port profile in the module. Sometimes more functionality is added but often it is not the right profile, or not the exact profile that is required and the small MCU that it runs on is insufficient for good performance. For this reason, and the fact that HCI based modules and chips are far less expensive, the third option is preferred.

The current architecture of the second and third elements in this list are shown in the following diagram:

The profiles offered today off the shelf for Bluetooth are:

- Serial port profile (SPP)

- Hands free profile (HFP)
- Health data profile (HDP)

Note that Unison Bluetooth is priced with various profiles in a consistent manner with all Unison modules. This makes the Unison Bluetooth 2.1 offering one of the most cost effective solutions on the market.

Unison Bluetooth v2.1 Modules

Typical modules provide SPP or serial port profile in the module. Sometimes other profiles are provided but often lack flexibility and features when used like this. In addition performance is often poor because the tiny MCU in the module is doing too much work. The list below is a set of modules that are supported using SPP and HCI. Note that any HCI module will work off the shelf in a few minutes.

- Panasonic 1315
- ST Ericson (consult factory)
- TI WL1271 HCI
- RoweBots CSR HCI module (pending release)

WiFi In Use With Unison

WiFi is the most commonly used wireless protocol used today for short distance communications. This first video shows WiFI used with USB mass storage class on a microcontroller to provide access to a web server via a smart phone.

WiFi is also expected to be widely using in the Internet of Things or if you prefer, Machine to Machine communications. The next video explains how it is more generally used for a variety of purposes.

WiFi is generally the wireless network of choice where high bandwidth is required, coverage needs to be localized and power is not a concern. Today we all use WiFi to access files in a variety of home and business settings with little attention paid to setup and configuration. It is a slmple tool in our society today.

WiFi Design and Implementation

When talking about WiFi, we are actually talking about a range of protocols and communications specifications which span the last 1.5 decades. Today we have 802.11a, b, g, to 802.11n with a maximum of 600Mbits/second. New standards like 11ac and 11ad. The differences between these from a user point of view are not relevant with the exception of the security standards (more below).

The various data rates and improvements are achieved through larger QAM constellations (8X8 or 256 or a 1 byte qbit) , Orthogonal Frequency Division Multiplexing (ODFM) which uses multiple frequencies and sends multiple bits this way, Direct Sequence Spread Spectrum (DSSS) and combinations of these strategies. Date rates are expected to be pushed over 1Gbit/sec for 11ac and up to 7Gbits/sec for 11ad.

WiFi runs in the 2.4GHz ISM band, and 5GHz bands. Channels are spaced 5MHz apart in the 2.4GHz band. The 5GHz band is considerably more complex.

Security in the WiFi domain is quite important and the initial WEP security was easily broken. Newer versions such as WPA and WPA2 are much stronger and these security enhancements offer good protection today. In enterprise WiFi environments, typically a Radius client server model is used along with EAP-TLS support.

WiFi Integration With TCP/IPv4/IPv6

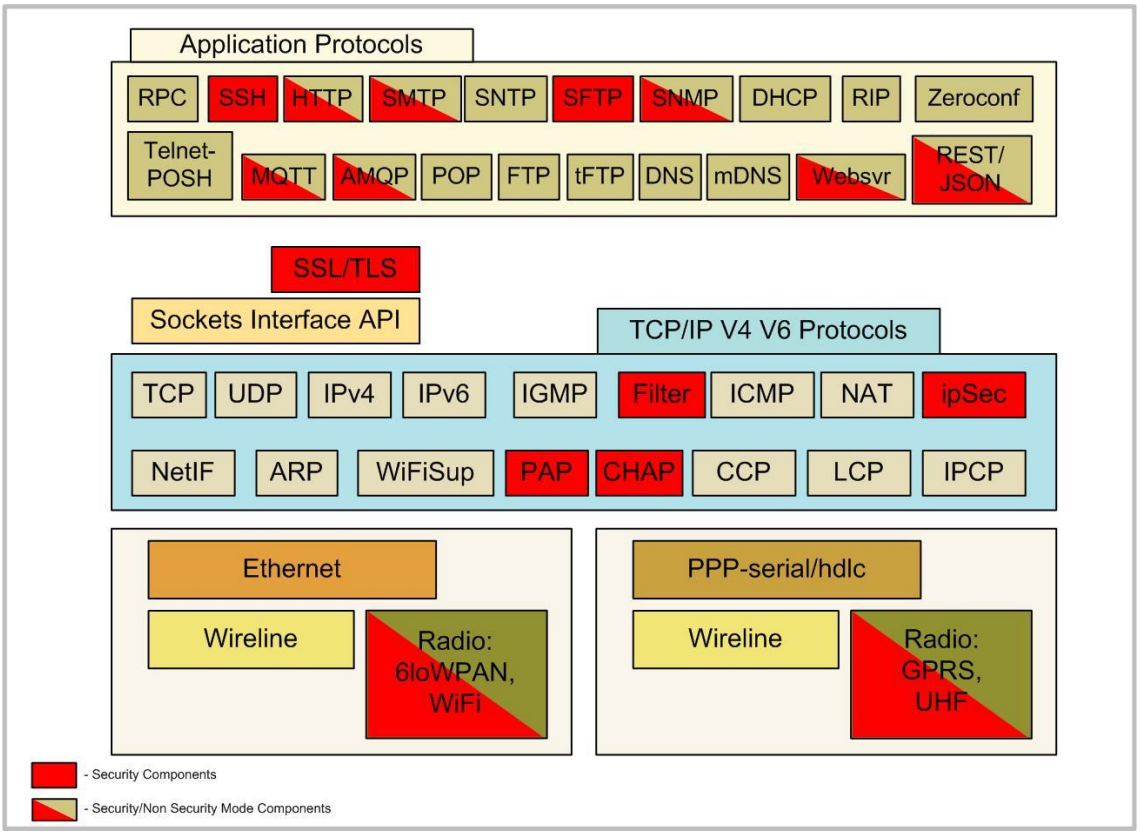

Figure 28. Protocols

At the bottom of the diagram you can see two possible collections of network interfaces. One set of interfaces uses PPP and a serial data stream to send packets over either a wireline or wireless network. This is shown on the right. The second interface on the left is for WiFi.

On the lower left of the diagram you can see there is a network interface consisting of both wireline and wireless connections like the right hand side. One of the wireless connections is WiFi. What this means is that as IP packets arrive or get sent out, they can be routed to/from this interface to the correct part of the system for processor or output as required. This selection of the WiFi interfaces takes care of the routing of packets out onto the WiFi radio and brings return packets into the protocol stack for processing.

Above this interface you can see a WiFiSupplicant. This piece of software handles the setup and processing of the WiFi with respect to security. As discussed above, it would make sure that the correct WPA and WPA2 security codes were used and the packet should be allowed to proceed into the stack.

As you can see from this diagram, the WiFi packets get processed and become part of the IPv6 and then the TCP packets as they proceed through the system layers. They also go through a TLS/SSL layer and on to higher level protocols. From this it should be clear that the WiFi implementation is tied direclty to the TCP/IPv4/IPv6 stack and that all higher level protocols run over this service.

WiFi Modules

Unison's WiFi support is fast and easy to use and works quickly and easily with a wide set of WiFi modules. It does not work with non-embedded systems modules simply because insufficient support is provided by the silicon and module vendors for these implementations. (This means that you are unlikely able to purchase a USB WiFi dongle and make it work in this environment.)

Supported WiFi modules:

- RedPine
- Microchip

Zigbee

Zigbee offers low data rate, low power mesh networking protocols. Over 300 companies support Zigbee today.

Zigbee really comes in three different flavors today. The first is standard Zigbee based on the second standard. The next one is Zigbee Pro which is based on the second standard and is backward compatible with Zigbee. The third flavor is Zigbee IP which utilizes 6loWPAN, TCP/IPv6 and RPL to extend Zigbee Pro ino the IPv6 networking domain.

ZigBee has a defined rate of 20-250 kbit/s, best suited for periodic or intermittent data or a single signal transmission from a sensor or input device. Applications include wireless light switches, electrical meters with in-home-displays, traffic management systems, and other consumer and industrial equipment that requires short-range wireless transfer of data at relatively low rates. The technology defined by the ZigBee specification is intended to be simpler and less expensive than other wireless networking technologies albeit with less performance and less power.

ZigBee operates in the industrial, scientific and medical ISM band bands (868 MHz in Europe, 915 MHz in the USA and Australia and 2.4 GHz worldwide). Data transmission rates vary from 20 kilobits/second at 868 MHz to 250 kilobits/second at 2.4 GHz . The network topologies can be either star, tree or mesh networks. The radio standard from 802.15.4 is used to support both the MAC and physical layers of the network.

Zigbee networks are secured by 128 bit encryption keys. This is typical for other wireless networks today. It offers good security at the present time but will need upgrades in the future as the keys are static.

Four main components are used to build up the Zigbee capabilities as shown in the diagram in a top down fashion. These components are:

- Application objects which are manufacturer specific;
- Zigbee device objects which record: device roles, management of requests to join a network, device discovery and security.
- Applications networking layers
- Networking layer

One of the main advantages of Zigbee has been the ability to go from standby to full operation in 30msec. This is substantially better than Bluetooth 2.1 or WiFi; however with Bluetooth 4.0 the new sensor standard creates a significant advantage for Bluetooth Smart (4.0).

The basic components of Zigbee are shown in the following diagram.

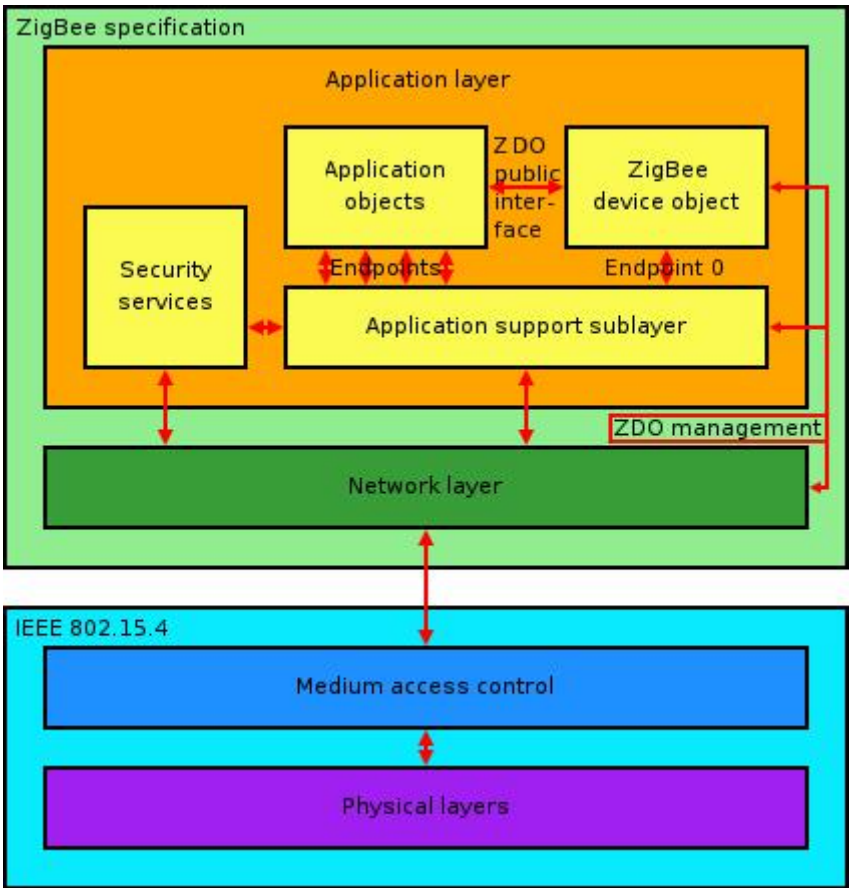

Figure 29. Components of Zigbee

RoweBots Unison RTOS offers three flavors of Zigbee. The first is an AT command or SPI version which interfaces with existing Zigbee modules. The recommended modules for this purpose are:

- Anaran (AIR) modules

The second type of solution is a standard Zigbee Pro solution using an off the shelf 802.15.4 radio. This version operates under direct programmatic control and runs on the microcontroller. This version is factory special order at the current time. The modules used for this include:

- Microchip 802.15.4 radio module
- Texas Instruments 802.15.4 radio module
- Anaren AIR modules

The third type of Zigbee support available is Zigbee IP. This solution can be based on either an Anaren AIR module running Zigbee or the Zigbee PRO solution running on the microcontroller.

GSM/GPRS, 3G, LTE, GPS, M2M

GSM/GPRS, 3G, LTE, GPS, M2M are all the telecommunications main protocols to implement their view of M2M or machine to machine communications. A broader view of M2M would include wireline Internet, WiFi, Bluetooth and Zigbee with networks of wireless sensors. Others see this later implementation as the "Internet of things" or IoT. In this context we will use the telecommunications firms definition for M2M and the telecommunications wireless solutions.

Using standard wireless modems for GSM/GPRS, 3G, LTE and GPS full Internet connetivity with secure access can be added to any application. The basic approach is to run PPP to serialize TCP/IPv4/IPv6 data streams and to use TLS or IPSec to secure the end to end channel. Generally these modems have either an SDIO, SPI or UART interface and this basic operation requires command modes to make the initial connection followed by enabling PPP data communication. Often stay alive packets are required as well to ensure that the link stays up over time.

These modules are relatively easy to use and connect. SDIO offers better performance, UART offers the best ease of use and SPI is a mid-point compromise. The Unison RTOS software requires some trivial customization for control for specfic modules but 99% of this technology is off the shelf.

802.15.4

802.15.4 is an IEEE specification for low power and low data rate radios. ZigBee operates in the industrial, scientific and medical ISM band bands (868 MHz in Europe, 915 MHz in the USA and Australia and 2.4 GHz worldwide). Data transmission rates vary from 20 kilobits/second at 868 MHz to 250 kilobits/second at 2.4 GHz .

The network topologies can be either star, tree or mesh networks in Zigbee and proprietary systems that use 802.15.4. The radio standard from 802.15.4 supports both the MAC and physical layers of the network but does not define

the topology. The overlay software defines the topology. The radio itself defines point to point connections and star structures.

Typical proprietary protocols which have been ported to the Unison RTOS include:

- TI SimpliciTI
- Microchip MiWi

The standards based protocols which would use the 802.15.4 radios include the following:

- Zigbee
- Zigbee PRO
- Zigbee IP
- 6loWPAN without Zigbee

The frequency ranges supported by various modules are:

- 868.0-868.6 MHz: Europe, allows one communication channel (2003), extended to three (2006)
- 902-928 MHz: North America, up to ten channels (2003), extended to thirty (2006)
- 2400-2483.5 MHz: worldwide use, up to sixteen channels (2003, 2006)

The actual modules that the Unison RTOS works with are:

- Anaren AIR
- Microchip PICtail
- Texas Instruments

6loWPAN

6loWPAN is a wireless protocol which provides an IPv6 address and TCP/IPv6 to all nodes in a network using a low data rate 802.15.4 radio. The technologies involved are:

- TCP/IPv6 header compression and decompression

- IPv6 address creation
- Dynamic routing between network nodes
- Ad hoc networking between network nodes
- Static routing between network nodes

The version of 6loWPAN in the video uses static routing at the ipv6 level Because there is no dynamic routing then nodes cannot be automatically added in an adhoc mesh network fashion as you might expect. The reason for this is related to the standardization efforts associated with RPL. Now RPL is agreed and approved, standardization can proceed in a more timely fashion

One of the key things that changed in the standardizaion was the introduction of Zigbee IP which uilizes IPv6 and 6loWPAN and RPL integrated with the lower levels of Zigbee to produce an integrated TCP/IPv6 - Zigbee solution. This will be available from RoweBots in the future.

The modules used for the implementation of 6loWPAN solutions either with raw use of the 802.15.4 radio or as part of a more comprehensive Zigbee IP solution include:

- Anaren AIR modules
- TI 802.15.4 radios
- Microchip 802.15.4 radios