Arduino meets Android

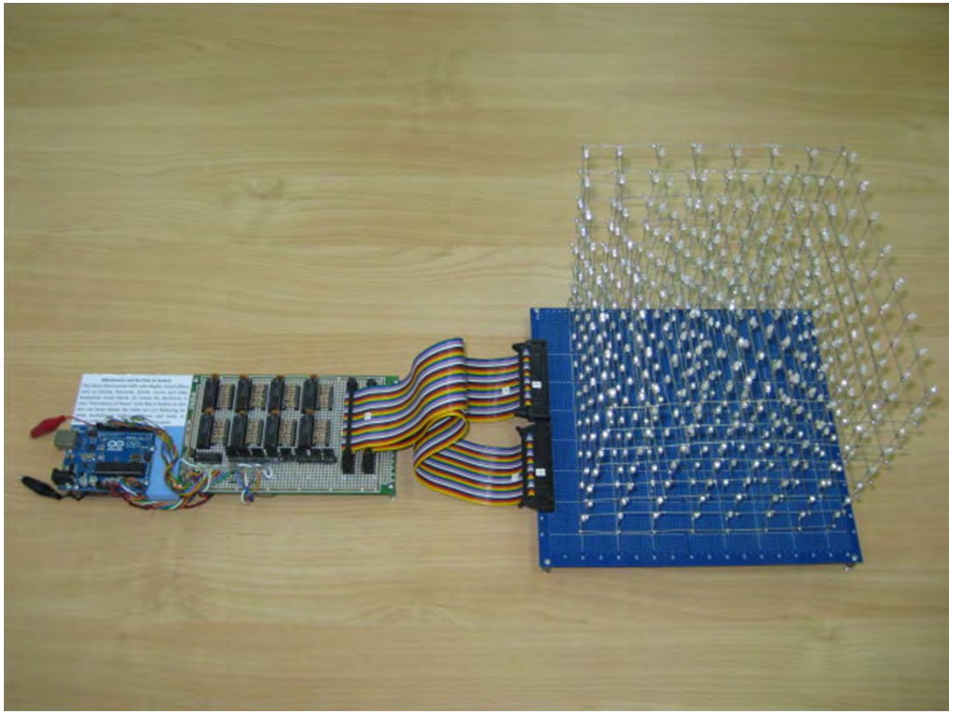
Creating Applications with
Bluetooth, Orientation Sensor, Servo, and LCD

Android + Arduino

- We will be learning
  - Bluetooth Communication
  - Android: built-in
  - Arduino: add-on board
- Android
  - Orientation sensor as input
- Arduino
  - Operate servo motor
  - Display orientation coordinate on LCD

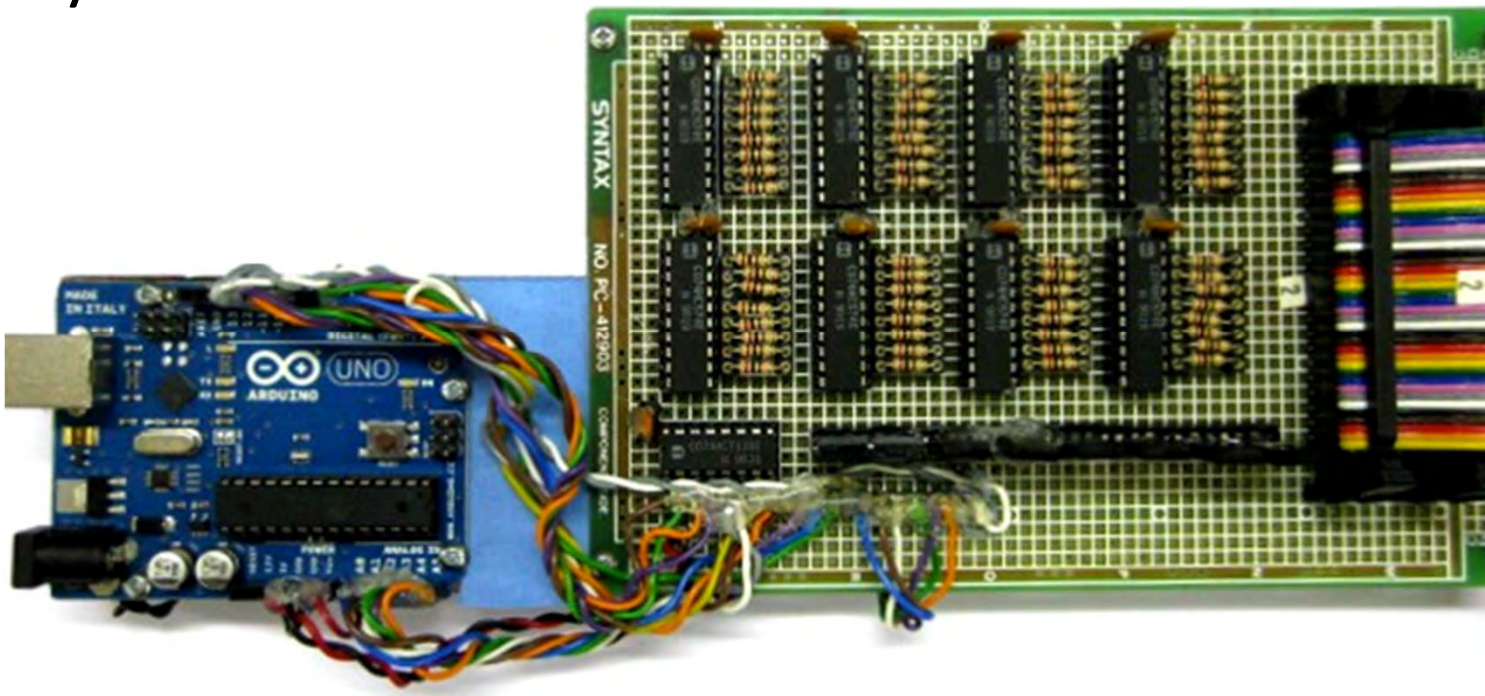
Multiplexing LEDs

- Arduino controls LEDs
- Visual Effects: Raining, Firework, Spirals, and others
- Created for EEE outreach to JC and Poly students
- Inspire learning interest for more students to NTU



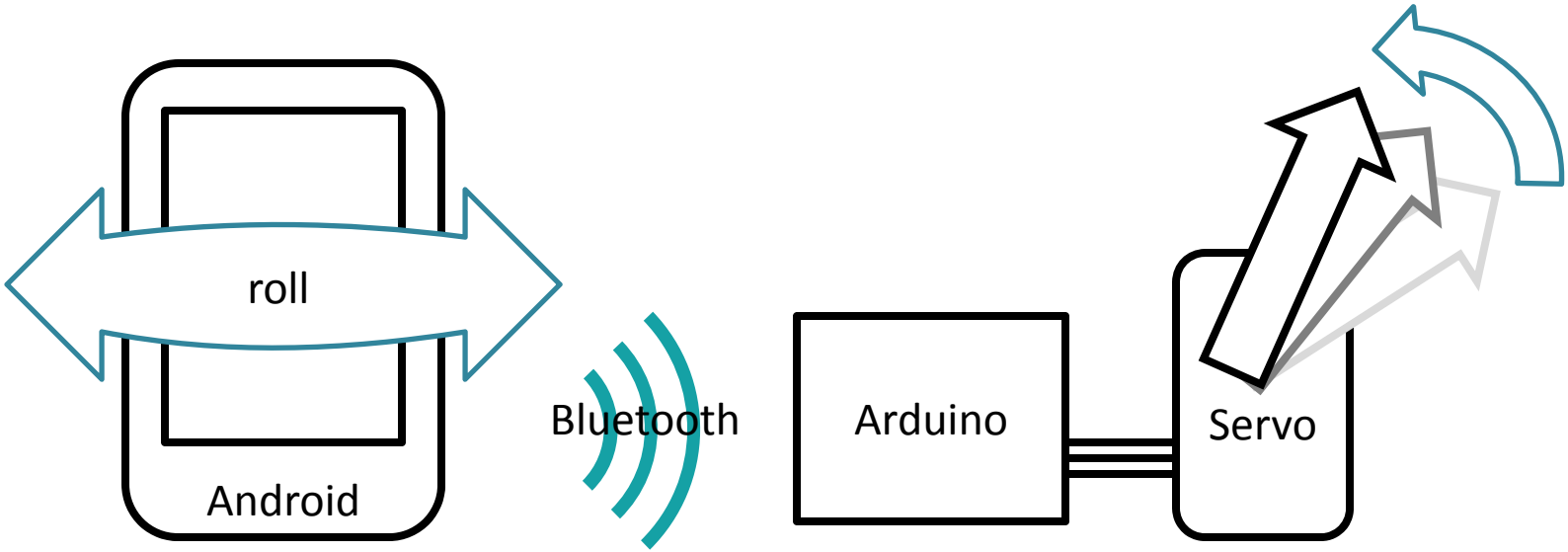
Multiplexing LEDs

- Arduino has limited number of output pins
- Wired up to Shift Registers , MOSFETs
- Persistence Of Vision
  - Image persist a moment of time on retina after LED off
- 1/8 duty cycle



Bluetooth + Orientation Sensor DEMO

- Servo follows the roll angle of phone



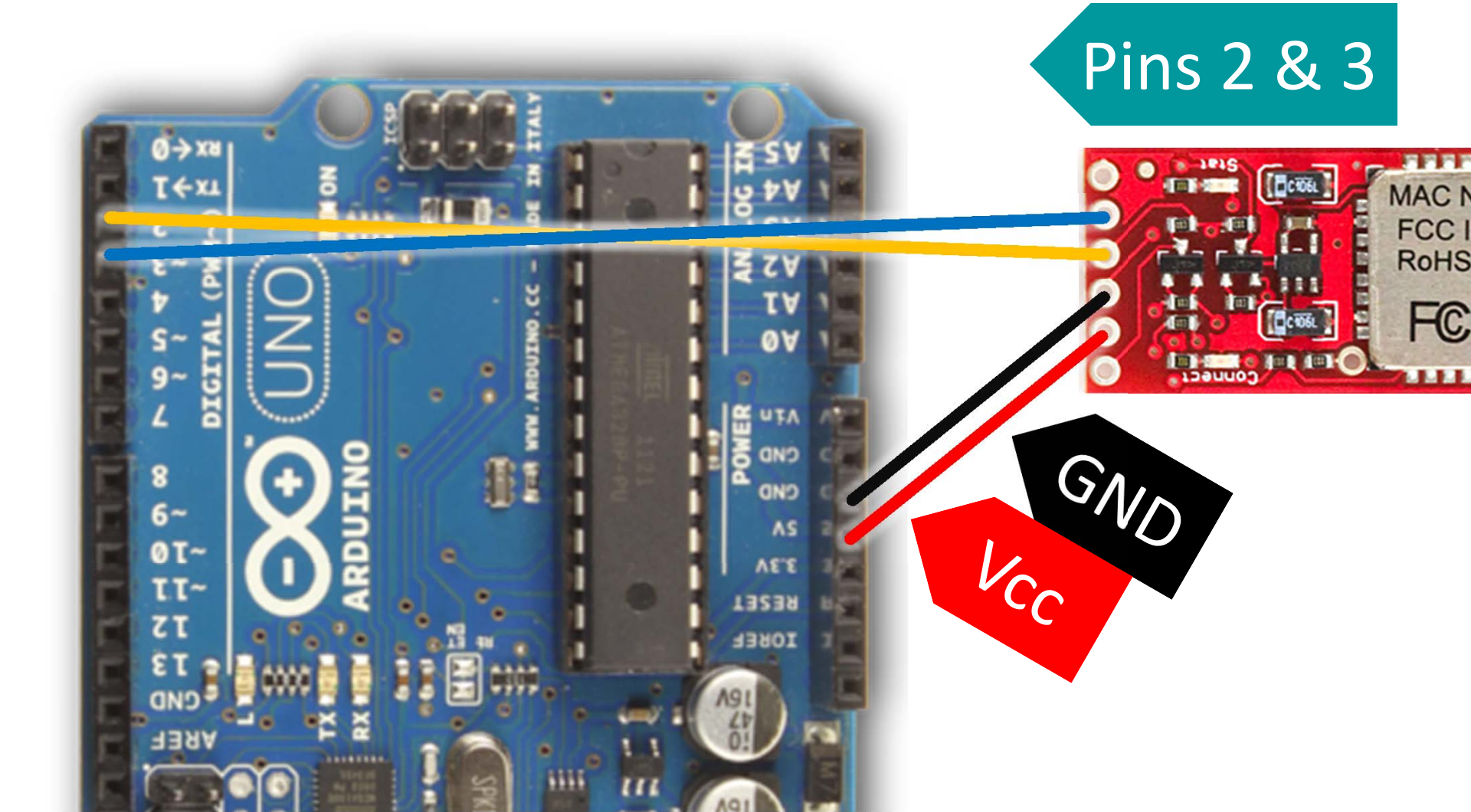
- Liquid Crystal Display to be added later
  - To display the angle

Bluetooth: FireFly

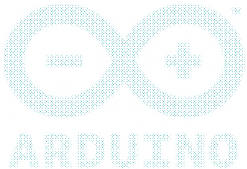
- BlueSMiRF Gold (FireFly)
- Encrypted connection
- Extremely small - 0.38 x 1.52 x 4.82 cm
- Distance up to 106 m at open air !

Firefly on Arudino

- Four connections



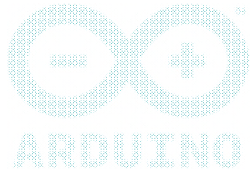
Android Phone



- Make sure
  - Bluetooth is enabled
  - Phone debugging turned ON
- Type 1234 to pair with FireFly

- Programming
  - `android.bluetooth` package

Bluetooth (Android)



- Android connects Arduino via the android.bluetooth package

```
btInterface = BluetoothAdapter.getDefaultAdapter();
```

- Obtain a list of paired devices with a reference to the adapter.

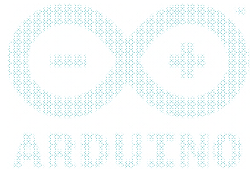
```
pairedDevices = btInterface.getBondedDevices();
```

- Search our firefly bluetooth from a list of devices

```
Iterator<BluetoothDevice> btlist = pairedDevices.iterator();
while (it.hasNext()){
        BluetoothDevice bd = it.next();
        if(bd.getName().equalsIgnoreCase(BluetoothName))
```

Get Adapter → Get Paired Devices → Search FireFly

Bluetooth (Android)



- After Bluetooth is found, initiate connection with the function

```
connectToBluetooth(bd);
```

- When connecting to the Bluetooth, createRfComSocketToServiceRecord() method is used

```
socket = bd.createRfcommSocketToServiceRecord
(UUID.fromString(
"00001101-0000-1000-8000-00805F9B34FB"));
```

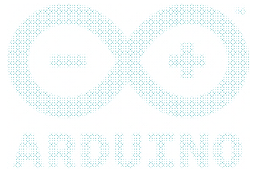
Universally Unique Identifier

- Connect to the socket by using the connect() method.

```
socket.connect();
```

Initiate Connection → Create Socket → Connect Socket

Bluetooth (Android)



- Monitoring the Bluetooth connection
- To do this, register two Bluetooth events:

```
ACTION_ACL_CONNECTED and ACTION_ACL_DISCONNECTED
```

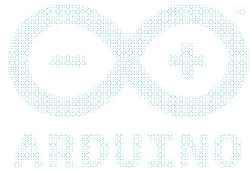
- When the device are connected, the handleConnected() method is invoked.

```
"android.bluetooth.device.action.ACL_CONNECTED"))
{ handleConnected();
```

- When the remove device disconnects, the handleDisconnected() method is invoked.

```
"android.bluetooth.device.action.ACL_DISCONNECTED"))
{ handleDisconnected();
```

Bluetooth (Android)



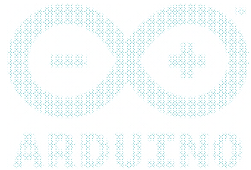
- *Once* `handleConnected()` *is* invoked, the connection is established.
- Set up the input and output streams for communication between Android and Android

```
is = socket.getInputStream();
os = socket.getOutputStream();
```

- Error Event
  - Call close method of the socket to disconnect

```
socket.close();
```

Bluetooth Permission



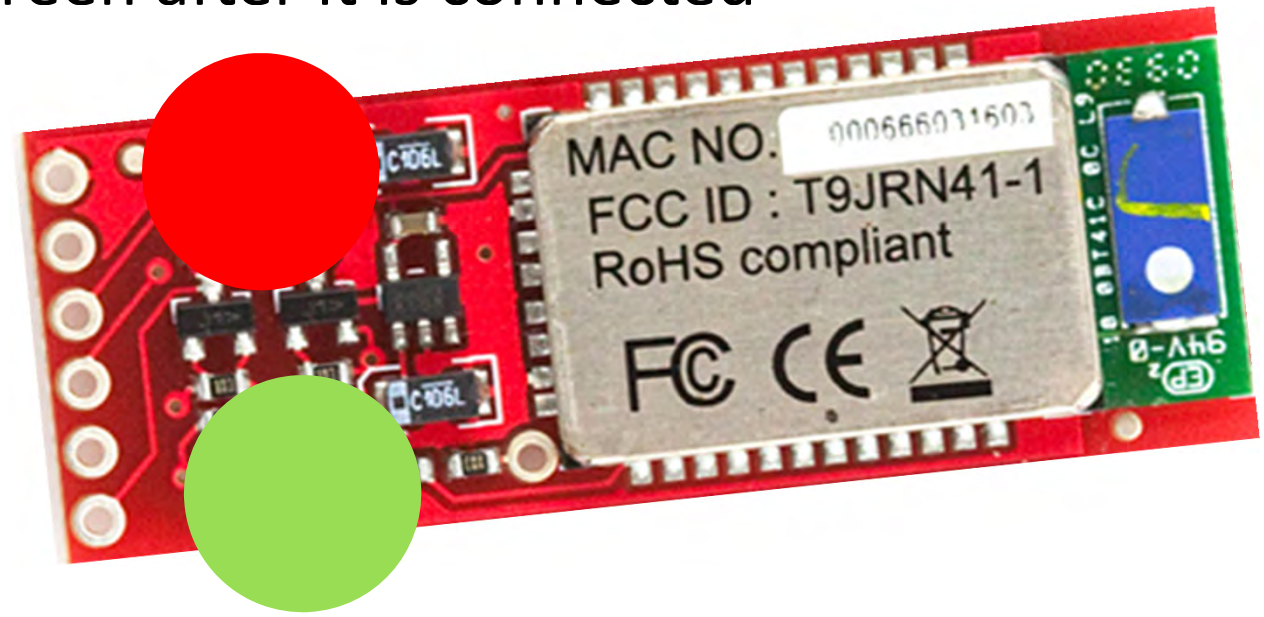
IMPORTANT

- Defined Permission in **AndroidManifest.xml** file:

  <uses-permission android:name=
  "android.permission.BLUETOOTH">
  </usespermission>

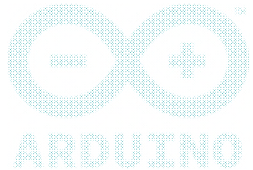
Bluetooth Keypoints

- LEDs turn green after it is connected



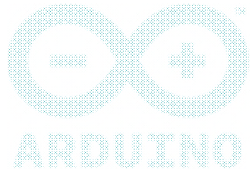
- Set BluetoothName to your purchased bluetooth
- Likewise for UUID

Android BT Summary



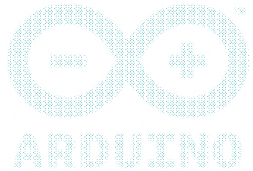
- Pair devices first
- Search our bluetooth among the Paired devices
- Connect with `createRfComSocketToServiceRecord()`
- UUID (Universally Unique Identifier)
- Socket Connection
- Events : `ACTION_ACL_CONNECTED and ACTION_ACL_DISCONNECTED`
- Input and Output Streams
- Transmit data through output stream
- Android Bluetooth permission

Android Sensors



- *Sensor.TYPE_ACCELEROMETER*
  - Measures acceleration in three dimensions
- *Sensor.TYPE_GYROSCOPE*
  - Gyroscope
- *Sensor.TYPE_LIGHT*
  - Ambient light sensor
- *Sensor.TYPE_MAGNETIC_FIELD*
  - Measures magnetic field compass
- *Sensor.TYPE_ORIENTATION*
  - Measures orientation in three dimensions
- *Sensor.TYPE_PRESSURE*
  - Measures pressure
- *Sensor.TYPE_PROXIMITY*
  - Measures distance the phone is away from another object, such as your ear
- *Sensor.TYPE_TEMPERATURE*
  - Measures ambient temperature

Orientation Sensor



- In the `android.hardware` package, the SensorManager class has the activities that we need.

- Obtain a reference to the SensorManager

```
SensorManager sManager = (SensorManager)
    getSystemService(Context.SENSOR_SERVICE);
```

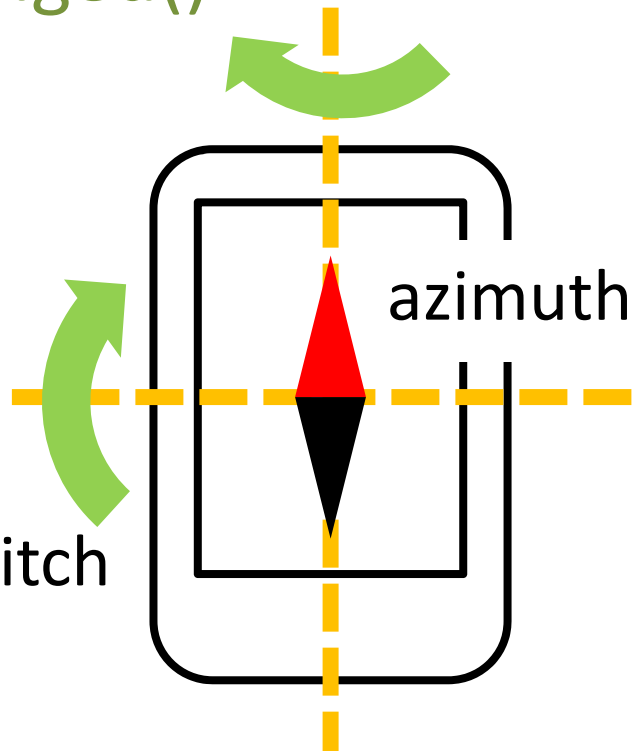
- To use the Orientation Sensor, call

```
Sensor orientationSensor =
    sManager.getDefaultSensor(Sensor.TYPE_ORIENTATION);
```

Orientation Sensor

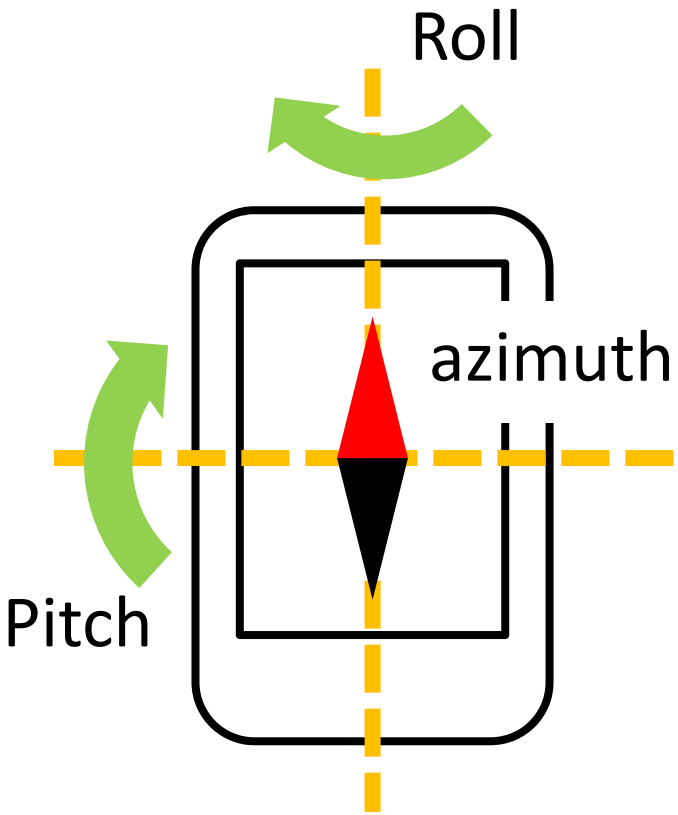
- Value is read from a sensor by implementing `SensorEventListener` interface.
- `SensorEvent` instances are then sent to a method named `onSensorChanged()`

```
public void onSensorChanged
    (SensorEvent event) {
    try {
        azimuth = event.values[0];
        pitch = event.values[1];
        roll = event.values[2];
```

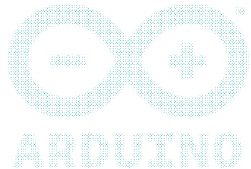


Orientation Sensor

- Three events
  - Aziumuth (z-axis)
    - angle in current reference to Magnetic North
    - 0 <= azimuth < 360
    - 0 = North, 90 = East, 180 = South, 270 = West
    - `event.values[0];`
  - Pitch (x-axis)
    - tilted forwards or backwards
    - -180 <= pitch <= 180
    - `event.values[1];`
  - Roll (y-axis)
    - Rotate in relation to the bottom left hand corner of the screen
    - -90 <= roll <= 90
    - `event.values[2];`



Bluetooth



- Obtained orientation data
- Send the data through the output stream

```
byte[] sendData = data.getBytes();
                        // convert data to byte

os.write(sendData);
os.flush();
                  // house keeping to
                  // flush out the output stream
```

Bluetooth

- SoftwareSerial Library

```
#include <SoftwareSerial.h>
```

- Initialize rxPin, txPin for the Bluetooth

```
SoftwareSerial bluetooth(rxPin,txPin);
```

- Setup the baud rate

```
bluetooth.begin(baudrate);
```

- Check for incoming data to bluetooth

```
if (bluetooth.available())
```

- Read data from bluetooth

```
byte data=bluetooth.read();
```

Servo

- Three wires

  Red - Vcc

  Black or Brown - Gnd

  Yellow or Orange or White – Connect to Arduino

Servo Implementation

- Servo Library

```
#include <Servo.h>
```

- Declare myServo object

```
Servo Servo1;
```

- Setup Pin Number

```
Servo1.attach(pin);
```

- Specify the angle in degrees to drive

```
Servo1.write(degree);
```

Bluetooth + Servo

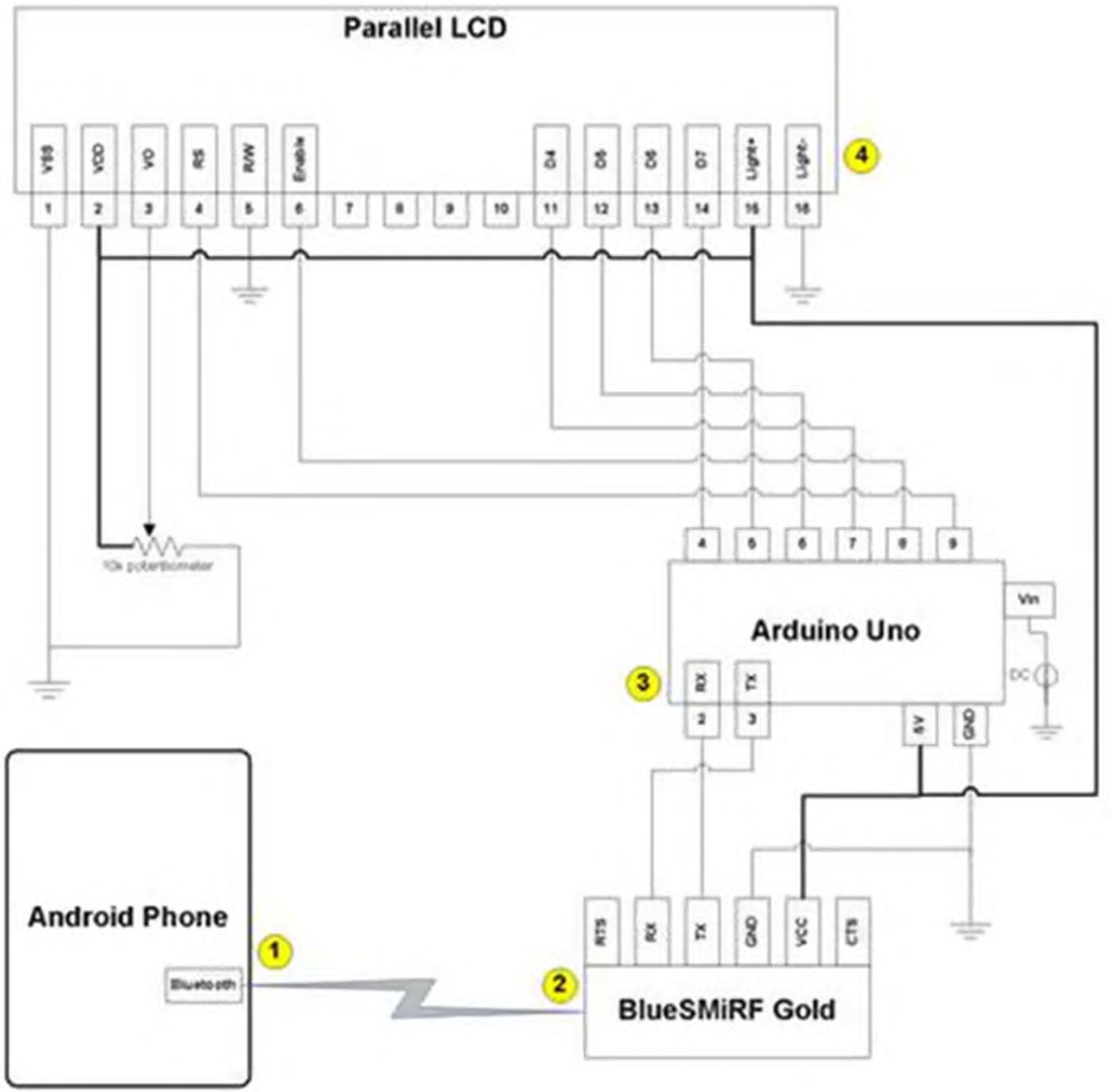

```
// Arduino receive one bit at a time in sequence

if (bluetooth.available()){ // check for bluetooth data
  data=bluetooth.read();    // read
  deg[ct]=(char)data;       // fill in the degree array
  ct++;
  if (data==0x2E){                  // search for dot "."
    Servo1.write(90+atoi(deg));  // move servo to deg
    ct=-1;                          // reset array
  }
 }
```

Liquid Crystal Display

- LCD Screen (compatible with Hitachi HD44780 driver)
- 10k Potentiometer to control the light intensity

Wiring of LCD



LCD Implementation

- LCD library

```
#include <LiquidCrystal.h>
```

- Setup the Pins for the LCD

```
LiquidCrystal lcd(9, 8, 7, 6, 5, 4);
```

- Setup the columns and rows of the LCD

```
lcd.begin(16, 2);
```

- Clear and set cursor to the beginning

```
lcd.clear();
lcd.setCursor(0, 0);
```

- Display Data on LCD

```
lcd.write(data);
```

Summary of Workshop

- Arduino wiring with Bluetooth
- Connect Android and Arduino *android.bluetooth.BluetoothAdapter*
- Azimuth, Pitch, Roll
- Arduino wiring with Servo and LCD
- Android Orientation drives Arduino Servo
- Display Android Orientation on Arduino LCD

Contact

In case, anyone need help

Mr. Yap Sing Loon
ESLYAP@NTU.EDU.SG