임베디드컴퓨팅
Embedded Computing
(0009488)
Bluetooth, App Inventor

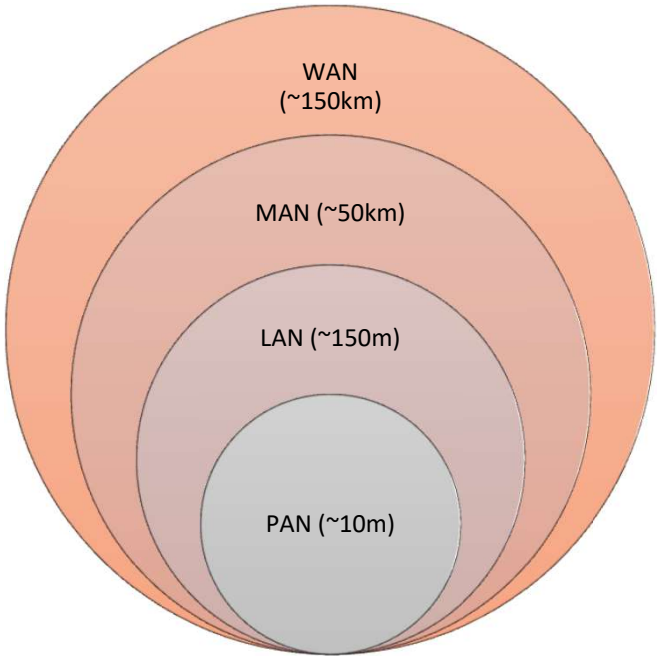
2022년 2학기

정보기술대학 정보통신공학과
김 영 필
ypkim@inu.ac.kr

INU 인천대학교
INCHEON NATIONAL UNIVERSITY

Bluetooth

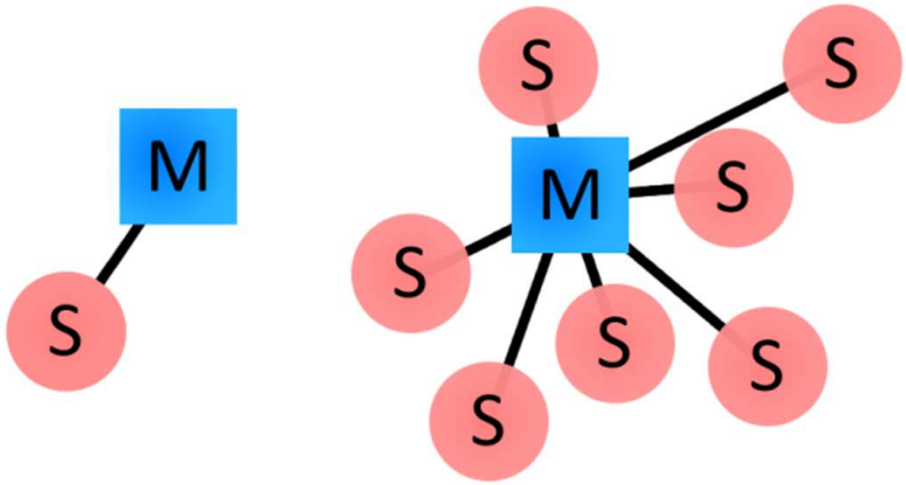
- A short-range wireless technology standard
  - IEEE
- Used for exchanging data between fixed and mobile devices over short distances
  - Personal Area Networks (PANs)
    - (BT5: 40 – 400m)
- Using **UHF radio waves** in the ISM bands
  - UHF: 300 Mhz – 3 GHz
  - ISM ITU Type B: 2.4 GHz – 2.5 GHz
  - BT: 2.402 GHz - 2.48 GHz



Radio spectrum reserved internationally for purposes, excluding applications in telecommunications.

Bluetooth communication model

- Bluetooth networks are referred to as Piconet
- [redacted] model
  - A single master device can be connected to up to seven different slave devices.
  - Any slave device in the piconet can only be connected to a single master



Ref - https://learn.sparkfun.com/tutorials/bluetooth-basics/all

Bluetooth Addresses and Names

- A Bluetooth device has a unique address (BD_ADDR)
  - Presented in a 12-digit hexadecimal value
  - OUI (24 bits) + Unique address (24 bits)
    - Organization Unique Identifier (OUI) identifies the manufacturer
- Can have user-friendly name for easy identification
  - Up-to-by 248 bytes long

Ref - https://learn.sparkfun.com/tutorials/bluetooth-basics/all

Connection Process (1)

- **Inquiry**
  - Initially, a device must run an inquiry to discover the other.
  - One device sends out the inquiry request, and any device listening for such a request will respond with **its** , and possibly **its** and other information.

- **Paging (Connecting)**
  - The process of forming a connection between two Bluetooth devices.
  - Each device needs to know .

Ref - https://learn.sparkfun.com/tutorials/bluetooth-basics/all

Connection Process (2)

- **Connection**
  - After completing the paging process, it enters the connection state.
  - While connected, a device can have the following modes
  - **Active Mode:**
    - Regular connected mode, where the device is actively transmitting or receiving data.

[redacted] **Mode:**

- A power-saving (sleep) mode.
- Only listen for transmissions at a set interval (e.g. every 100ms).

[redacted] **Mode (master -> slave):**

- A temporary, power-saving mode for a defined period.
- The master can command a slave device to hold.

[redacted] **Mode (master -> slave):**

- The deepest of sleep modes.
- Parked slave will become inactive until the master tells it to wake back up.

Ref - https://learn.sparkfun.com/tutorials/bluetooth-basics/all

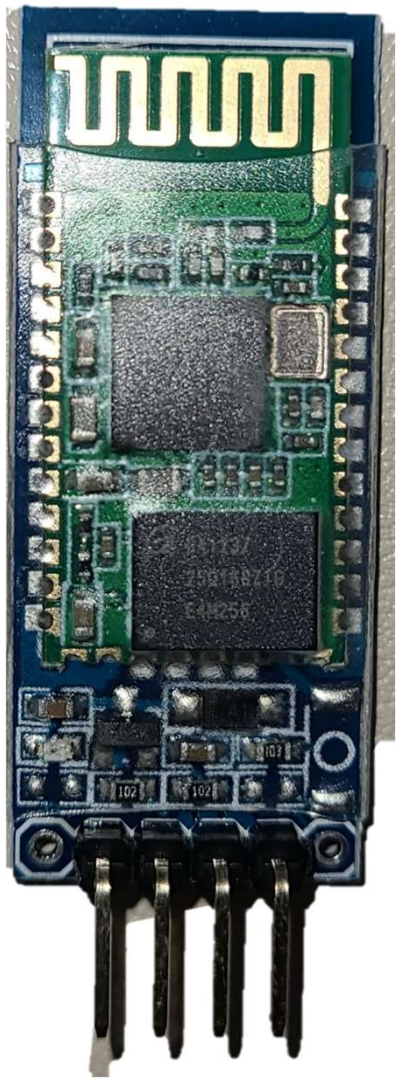
Comparison BT with others

Name	Bluetooth Classic	Bluetooth 4.0 Low Energy (BLE)	ZigBee	WiFi
IEEE Standard	802.15.1	802.15.1	802.15.4	802.11 (a, b, g, n)
Frequency (GHz)	2.4	2.4	0.868, 0.915, 2.4	2.4 and 5
Maximum raw bit rate (Mbps)	1-3	1	0.250	11 (b), 54 (g), 600 (n)
Typical data throughput (Mbps)	0.7-2.1	0.27	0.2	7 (b), 25 (g), 150 (n)
Maximum (Outdoor) Range (Meters)	10 (class 2), 100 (class 1)	50	10-100	100-250
Relative Power Consumption	Medium	Very low	Very low	High
Example Battery Life	Days	Months to years	Months to years	Hours
Network Size	7	Undefined	64,000+	255

Ref - https://learn.sparkfun.com/tutorials/bluetooth-basics/all

Bluetooth module

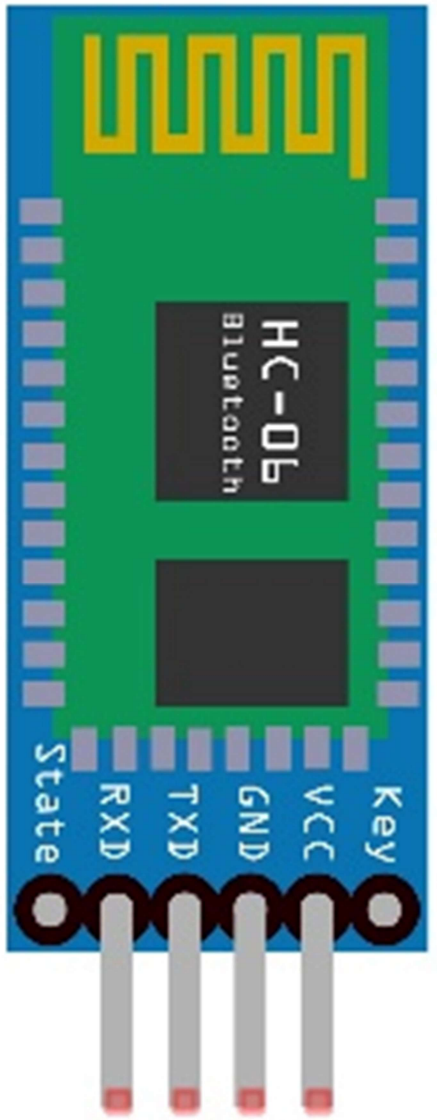
- Support Bluetooth 2.0 communication protocol
- Can only act as a
- For a master device, we use our smartphone.
- PIN number of each device should be matched for a valid connection.



HC-06

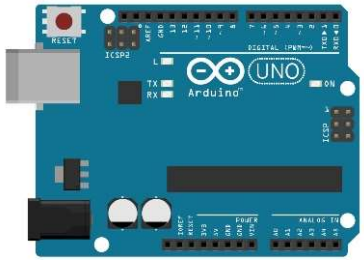
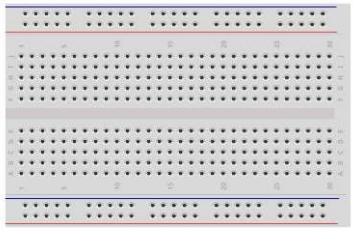
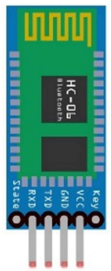
HC-06 pins

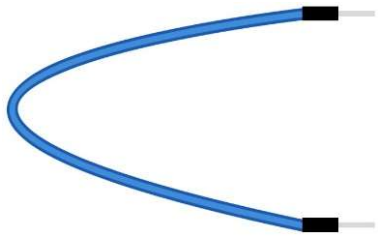
- The Bluetooth module uses four pins.
  - VCC: A pin that applies + power.
    - Operating voltage is 3.1 ~ 6.5V,
    - Connect it to the 3.3V or 5V pin of the Arduino.
  - GND: A grounding pin
    - connected to the GND pin of the Arduino.
  - TXD: A pin to transmit data.
    - Connect it to the pin set to RX on the Arduino.
  - RXD: A pin to receive data.
    - Connect it to the pin set for TX on the Arduino.



Lab: Bluetooth communication

- Let's connect the smart phone **(master)** and the Bluetooth module **(slave)** to transmit and receive data.
- Required H/W

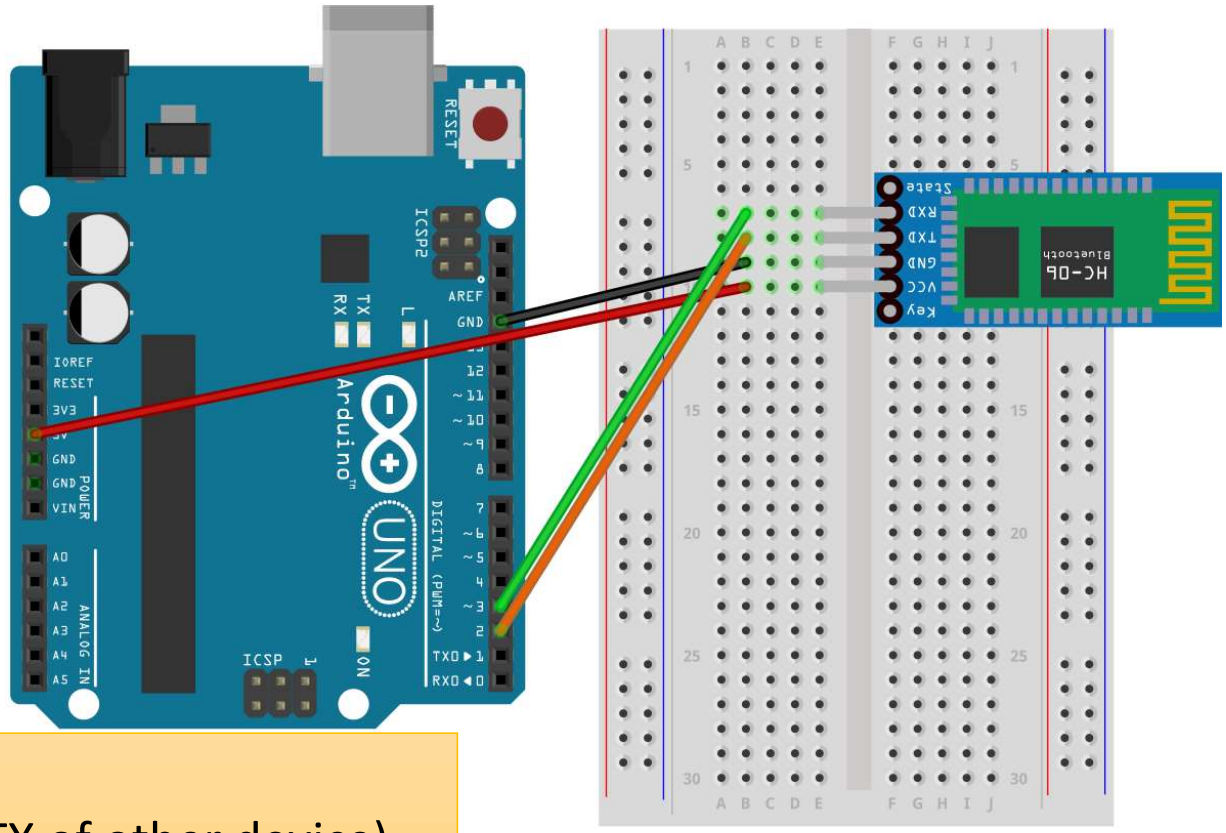
Arduino Uno board	Breadboard	Bluetooth module (HC-06)	Smartphone (Android / iOS)
			
x 1	x 1	x 1	x 1

Jumper cable (Male-Male)

x 4

Circuit wiring setup

- To send data to HC-06 (RXD), we use digital pin 3 (TX)
- To receive data from HC-06 (TXD), we use digital pin 2 (RX)

HC-06	Arduino pins
VCC	5V
GND	GND
TXD	digital 2 (RX)
RXD	digital 3 (TX)



The circuit:

- RX is digital pin 2 (connect to TX of other device)
- TX is digital pin 3 (connect to RX of other device)

SoftwareSerial Library

- Native serial communication in Arduino uses only digital pin 0 (RX) and 1 (TX) for UART (H/W)
  - UART can process communication tasks; thus, Atmega chip can do other work during data communication
- **SoftwareSerial library** allows to use other digital pins for serial communication
- **long serial buffer** is used for SoftwareSerial communication
- To use this library
  - #include<SoftwareSerial.h>
- Examples
  - https://www.arduino.cc/en/Tutorial/LibraryExamples/SoftwareSerialExample

Functions

- SoftwareSerial()
- available()
- begin()
- isListening()
- overflow()
- peek()
- read()
- print()
- println()
- listen()
- write()

Ref - https://www.arduino.cc/en/Reference/SoftwareSerial

SoftwareSerial (RX, TX, Inverse_logic)

- Used to create an instance of a SoftwareSerial object
  - The inverse_logic argument is optional (defaults to false)
  - Multiple SoftwareSerial objects may be created, however only one can be active at a given moment.
- Need to call SoftwareSerial.begin() to enable communication.
- Parameters
  - RX: the pin on which to receive serial data
  - TX: the pin on which to transmit serial data
  - inverse_logic: used to invert the sense of incoming bits
    - normal logic (default): LOW = 0 bit, HIGH 1 bit
    - inverse logic: LOW = 1 bit, HIGH = 0-bit

```
#include <SoftwareSerial.h>

const byte rxPin = 2;
const byte txPin = 3;

// set up a new serial object
SoftwareSerial mySerial (rxPin, txPin);
```

Ref. - https://www.arduino.cc/en/Reference/SoftwareSerialConstructor

SoftwareSerial: begin(speed)

- Sets the speed (baud rate) for the serial communication.
  - Supported baud rates:
    - 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 31250, 38400, 57600, 115200.
- Parameters
  - speed: the baud rate (long)
- Returns
  - none

```
void setup()  {
  // define pin modes for tx, rx:
  pinMode(rxPin, INPUT);
  pinMode(txPin, OUTPUT);
  // set the data rate for the SoftwareSerial port
  mySerial.begin(9600);
}
```

Ref. - https://www.arduino.cc/en/Reference/SoftwareSerialBegin

SoftwareSerial: available()

- Get the number of bytes (characters) **available for reading** from a software serial port.
  - **Data is already arrived** and stored in the **serial receive buffer**.
- Syntax
  - mySerial.available()
- Parameters
  - none
- Returns
  - the number of bytes available to read

```
void loop() {
  if (mySerial.available()>0){
  mySerial.read();
 }
}
```

Ref. - https://www.arduino.cc/en/Reference/SoftwareSerialAvailable

SoftwareSerial: read

- Return **a character** that was received on the RX pin of the software serial port.
  - Only one SoftwareSerial instance can receive incoming data at a time
    - select which one with the **listen()** function.
  - cf) **peek()** returns a character, also. What's the difference?
- Parameters
  - none
- Returns
  - the character read, or -1 if none is available

```
void loop()
{
  char c = mySerial.read();
}
```

Ref. - https://www.arduino.cc/en/Reference/SoftwareSerialRead

SoftwareSerial: write(data)

- Prints data to the transmit pin of the software serial port as raw bytes. Works the same as the Serial.write() function.
- Parameters
  - Serial.write() for details
- Returns
  - the number of bytes written, though reading that number is optional

```
SoftwareSerial mySerial(10, 11);

void setup()
{
  mySerial.begin(9600);
}

void loop()
{
  mySerial.write(45); // send a byte with the value 45

   int bytesSent = mySerial.write("hello"); //send the string "hello"
}
```

Ref. - https://www.arduino.cc/en/Reference/SoftwareSerialWrite

SoftwareSerial Limitation

- [redacted] even if using multiple software serial ports.
- Not all pins are available; it depends on your Arduino board
  - Available RX pins for Mega and Mega 2560
    - 10, 11, 12, 13, 14, 15, 50, 51, 52, 53, A8 (62), A9 (63), A10 (64), A11 (65), A12 (66), A13 (67), A14 (68), A15 (69).
  - Available RX pins for Leonardo and Micro
    - 8, 9, 10, 11, 14 (MISO), 15 (SCK), 16 (MOSI).
  - On Arduino or Genuino 101
    - The current maximum RX speed is 57600bps
    - RX doesn't work on Pin 13

Ref - https://www.arduino.cc/en/Reference/SoftwareSerial

Basic setup for Bluetooth slave

```
#include <SoftwareSerial.h>
#define RX 2
#define TX 3

SoftwareSerial mySerial

void setup() {
  Serial.begin(9600);
  mySerial.begin(9600);
}

void loop() {
  if (mySerial Serial.write(mySerial.read());
  if (Serial. mySerial.write(Serial.read());
}
```

Use library → Include library header files

Macro for functionality extension

SoftwareSerial object initialization

UART serial → 9500 baud
Bluetooth serial → 9600 baud

Bluetooth data has arrived → Send it to UART serial

UART serial writes data → Send it to Bluetooth slave

Smartphone as a master device

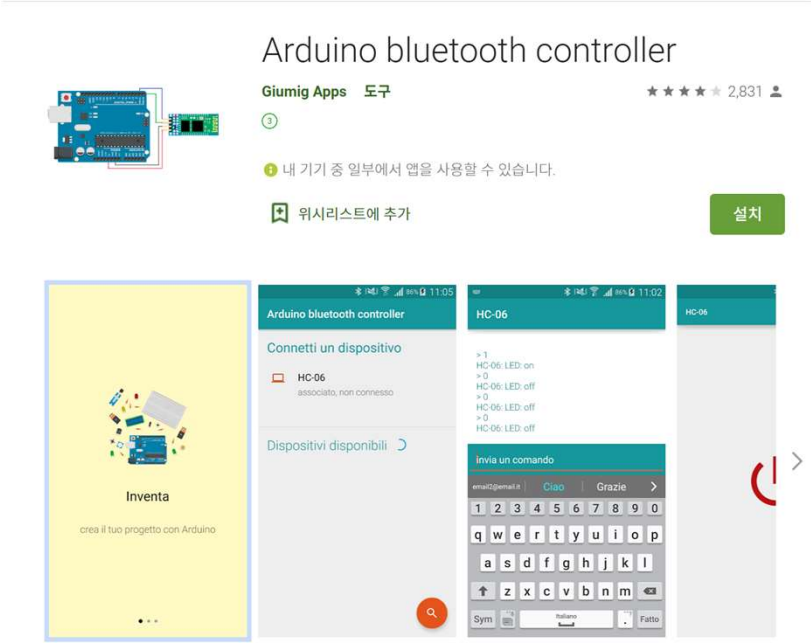
- For Android users
  - Two ways are possible
    - Install conventional Arduino Bluetooth controller App
      - Search "Arduino Bluetooth Controller" in Google Playstore
    - Build your own Bluetooth controller App
      - Using MIT App Inventor 2
- For iOS users
  - One way is possible
    - Install conventional Arduino Bluetooth controller App
      - Search "Arduino Bluetooth Controller" in Apple Appstore
    - App Inventor for iOS has not supported BT connectivity yet (2021.11)
      - http://doesappinventorrunonios.com/ (Check the status here)

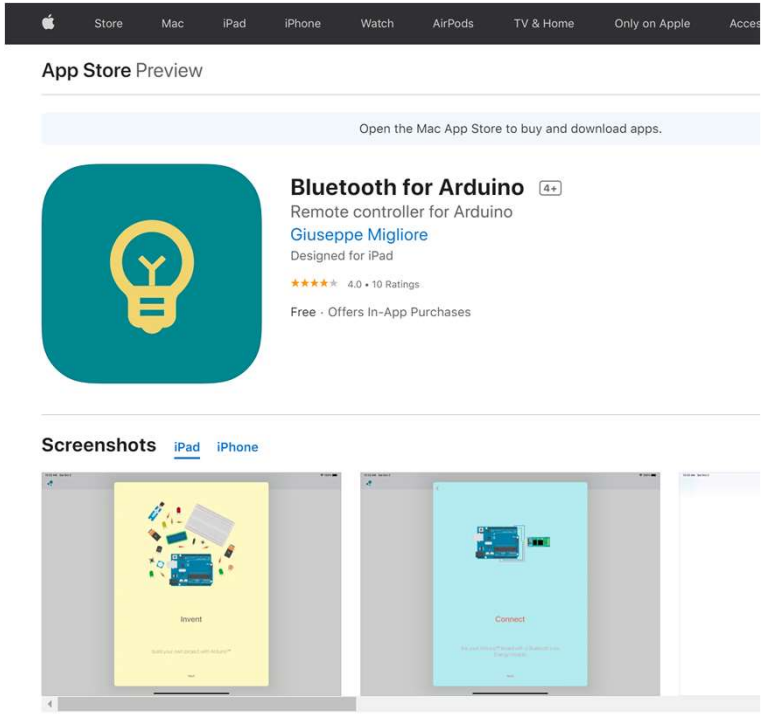
Bluetooth device pairing

- To test our Bluetooth slave sketch program, we need to prepare a Bluetooth master device
- We can use a smartphone (Android or iOS) and need to register our HC-06 using **a PIN** for Bluetooth communication
  - Default PIN value is **1234**
- After BT device registration, we can communicate with HC-06 via the Bluetooth controlling app

#1 Bluetooth controller app

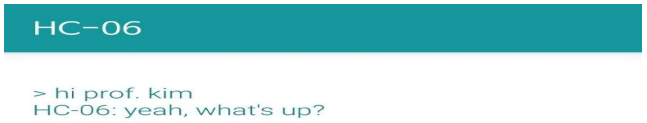
- Your smartphone can be a Bluetooth master device after installation of one of them
  - Bluetooth related or Location related permissions are required

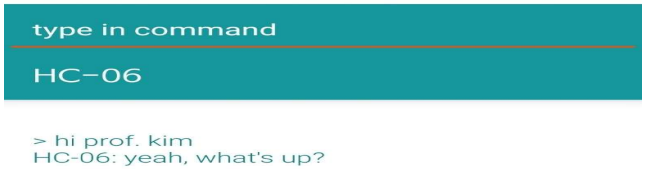




Bluetooth controller app

- Master device side
  - After HC-06 device pairing, Select "HC-06" in "Connect to a device" to connect
  - Select "Terminal mode" in "Connect in"
  - Send data in "type in command"
  - Receive data in terminal window





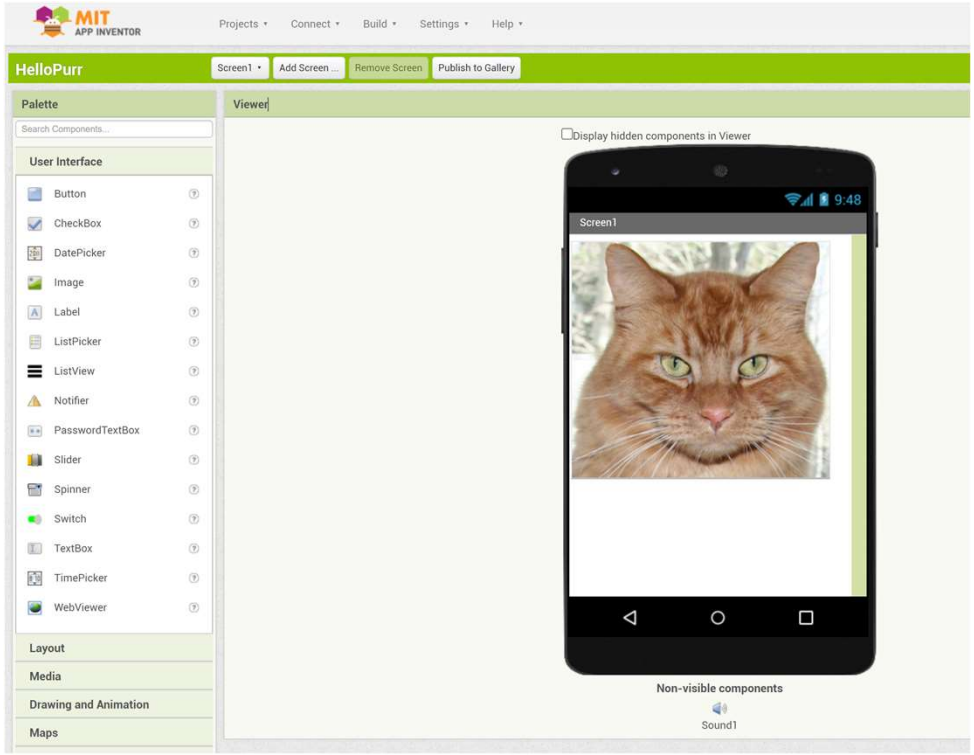
- Slave device side
  - Run Bluetooth slave sketch program
  - Open Serial Monitor
  - Send or Receive data

#2 App Inventor

- A [redacted] that allows you to easily create apps based on the Android or iOS platform.
  - Creating app = Exterior design (Designer) + programming (Block editor).
  - Cloud-based software service
  - Event-driven programming logic
- Features
  - **Easy design** of the appearance of your app using the **drag and drop** method.
  - **Easy programming** with **blocks** without knowledge of programming instructions (or statements).
  - **Easy handling of events or conditions** by block representations

MIT APP INVENTOR

https://appinventor.mit.edu/



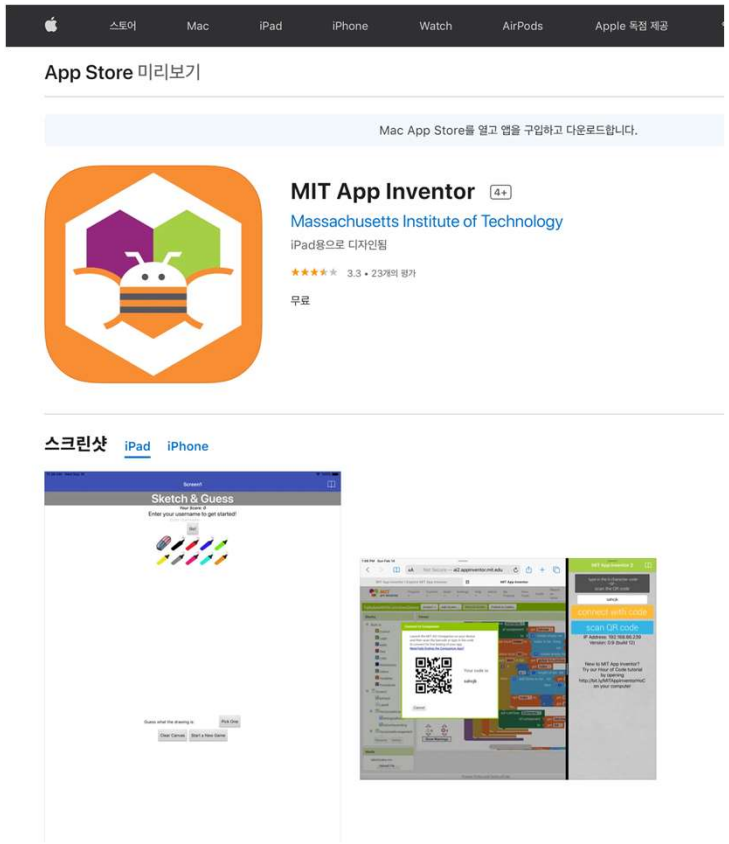
Getting started App Inventor with Smartphone

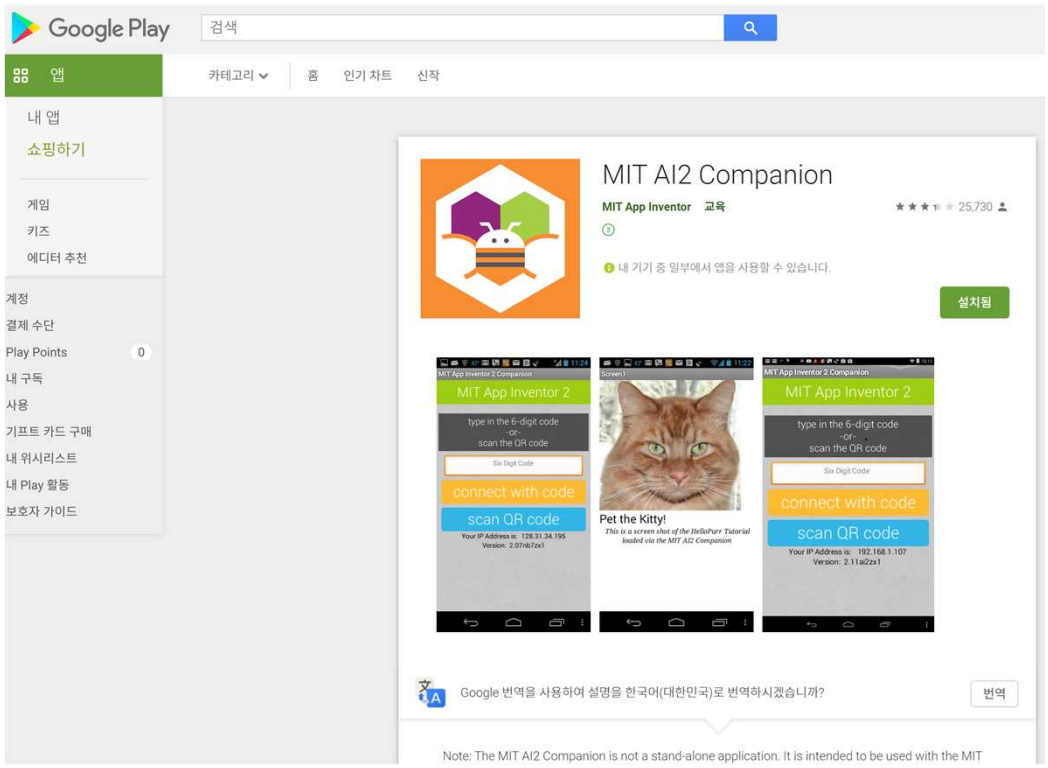
- Install **MIT App Inventor companion App** in your smartphone

For the detailed installation steps,
see https://appinventor.mit.edu/explore/get-started

Download on the App Store

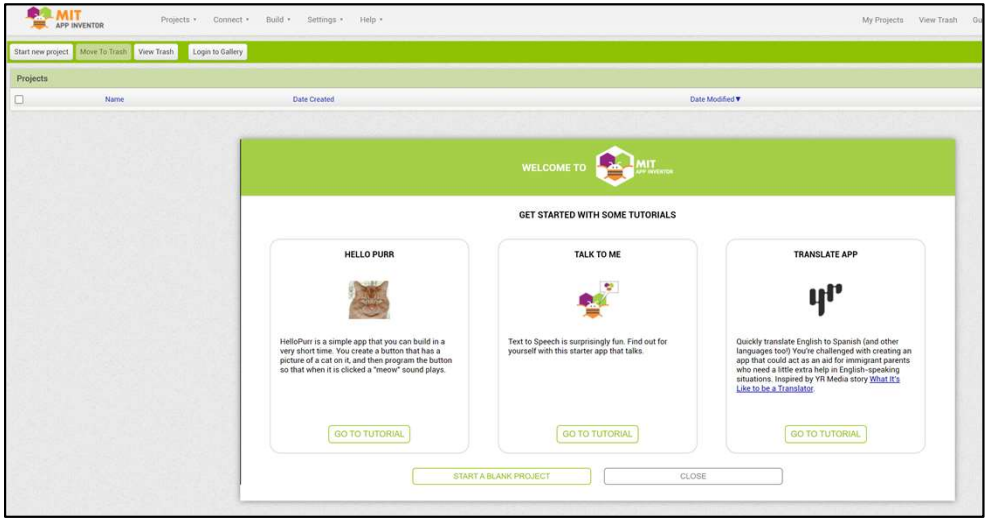
GET IT ON Google Play

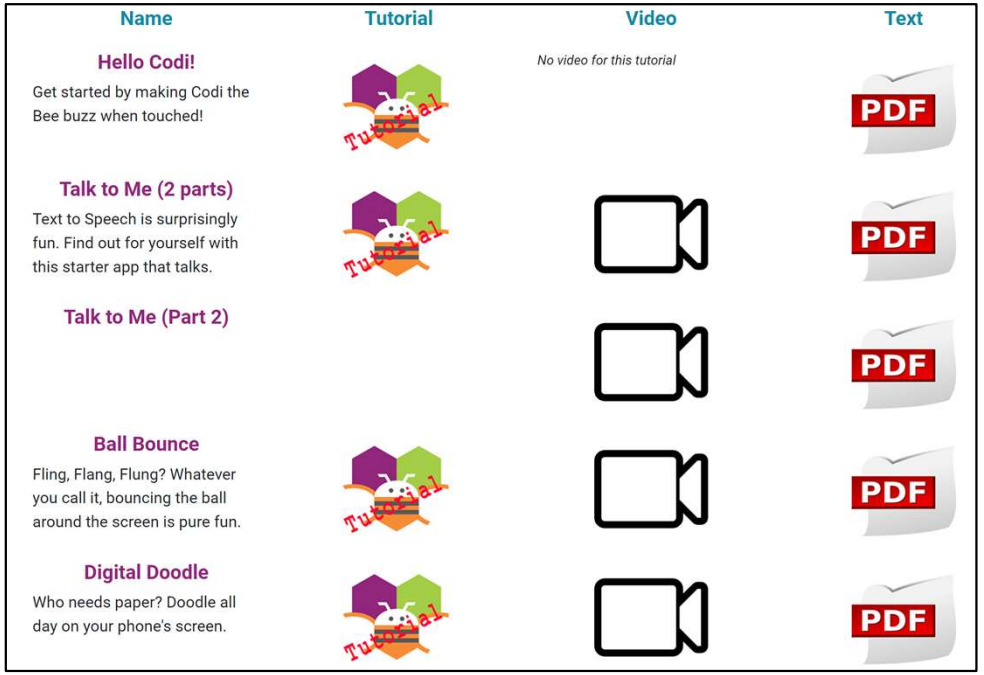




Access to App Inventor IDE

- You can use App Inventor IDE with or without an account
- Fastest way is just to click the link
  - http://code.appinventor.mit.edu
  - Login with your account, or
  - Login without account and use the auto-generated access code to revisit
- The tutorial materials are available (Video, Text, etc)
  - https://appinventor.mit.edu/explore/ai2/beginner-videos

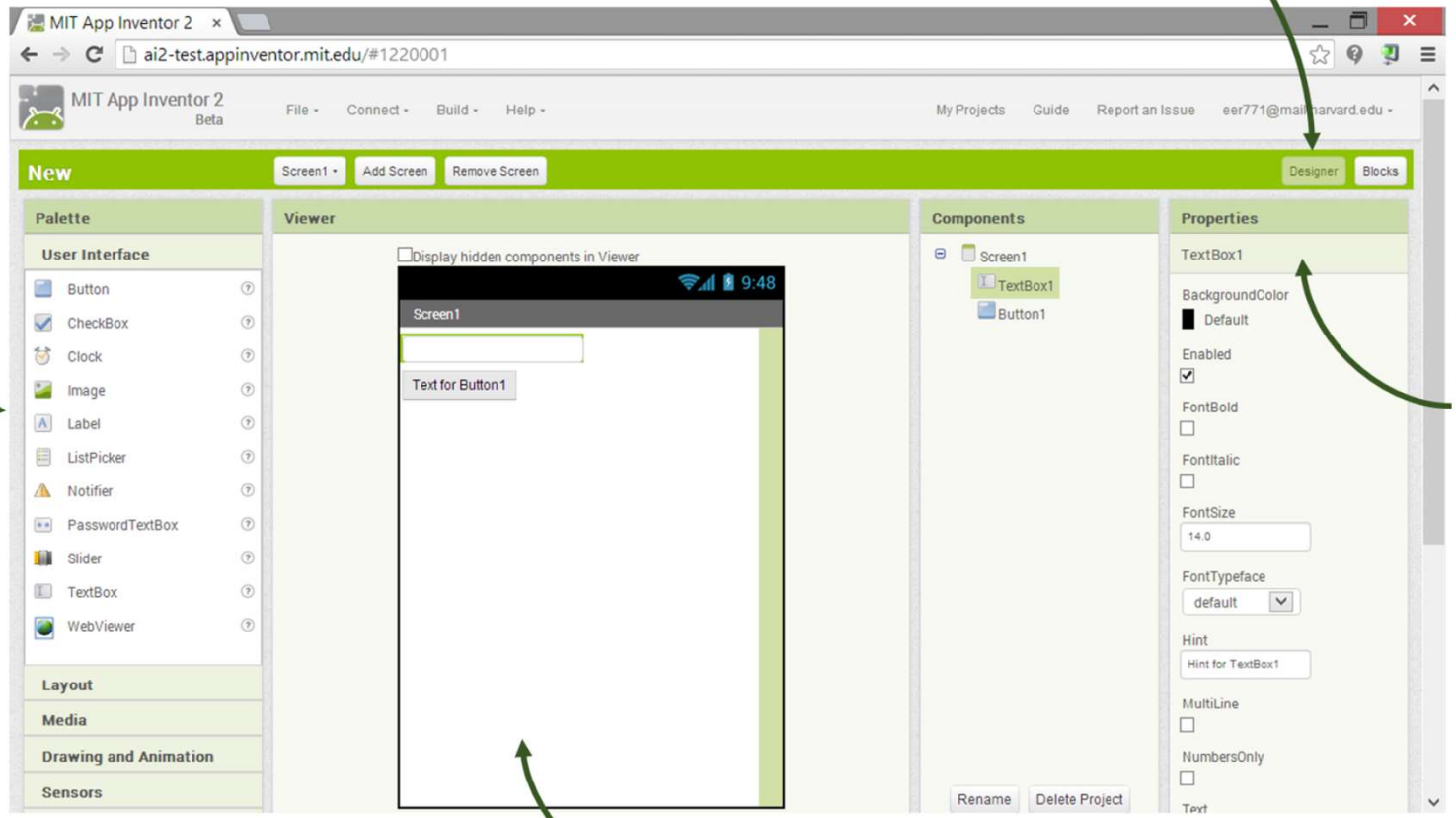




App Inventor Designer

Palette: Find your components and drag them to the Viewer to add them to your app.

Designer Button: Click from any tab to go to the Designer tab.



Properties: Select a Component in the Components List to change its properties (color, size, behavior) here.

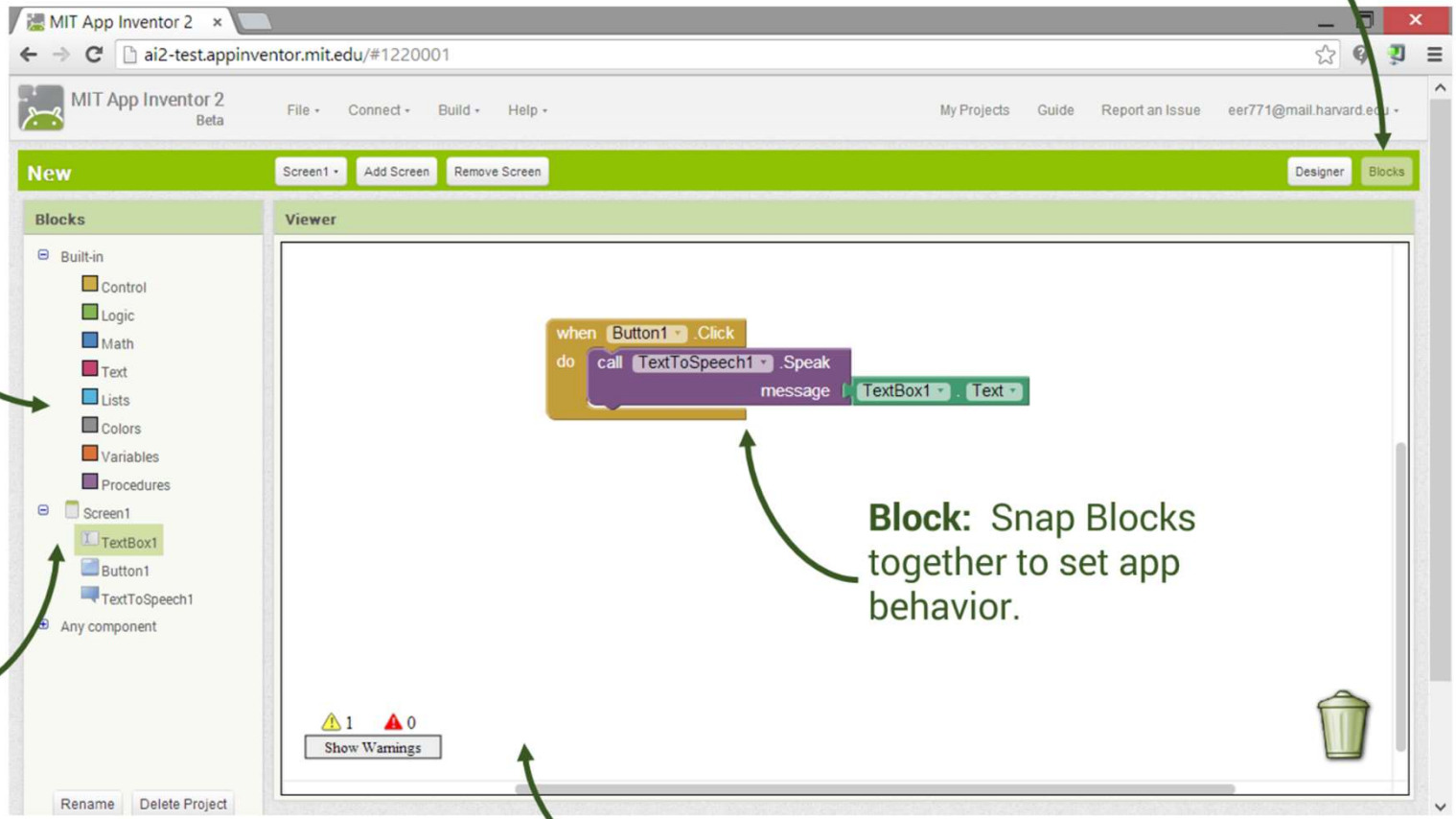
Viewer: Drag components from the Palette to the Viewer to see what your app will look like.

Ref - https://appinventor.mit.edu/explore/designer-blocks

App Inventor Blocks editor

Built-In Drawers: Find Blocks for general behaviors you may want to add to your app and drag them to the Blocks Viewer.

Blocks Button: Click from any tab to go to the Blocks tab.



Component-Specific Drawers: Find Blocks for behaviors for specific Components and drag them to the Blocks Viewer.

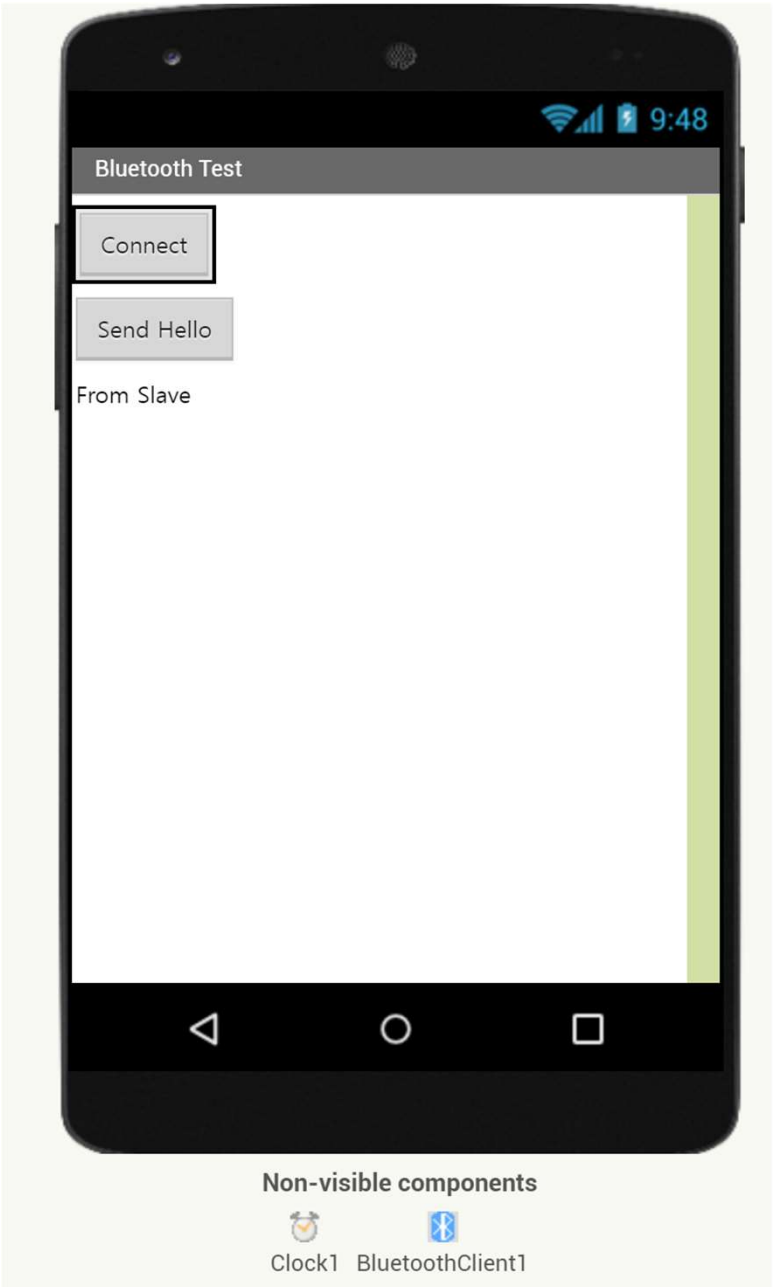
Block: Snap Blocks together to set app behavior.

Viewer: Drag Blocks from the Drawers to the Blocks Viewer to build relationships and behavior.

Ref - https://appinventor.mit.edu/explore/designer-blocks

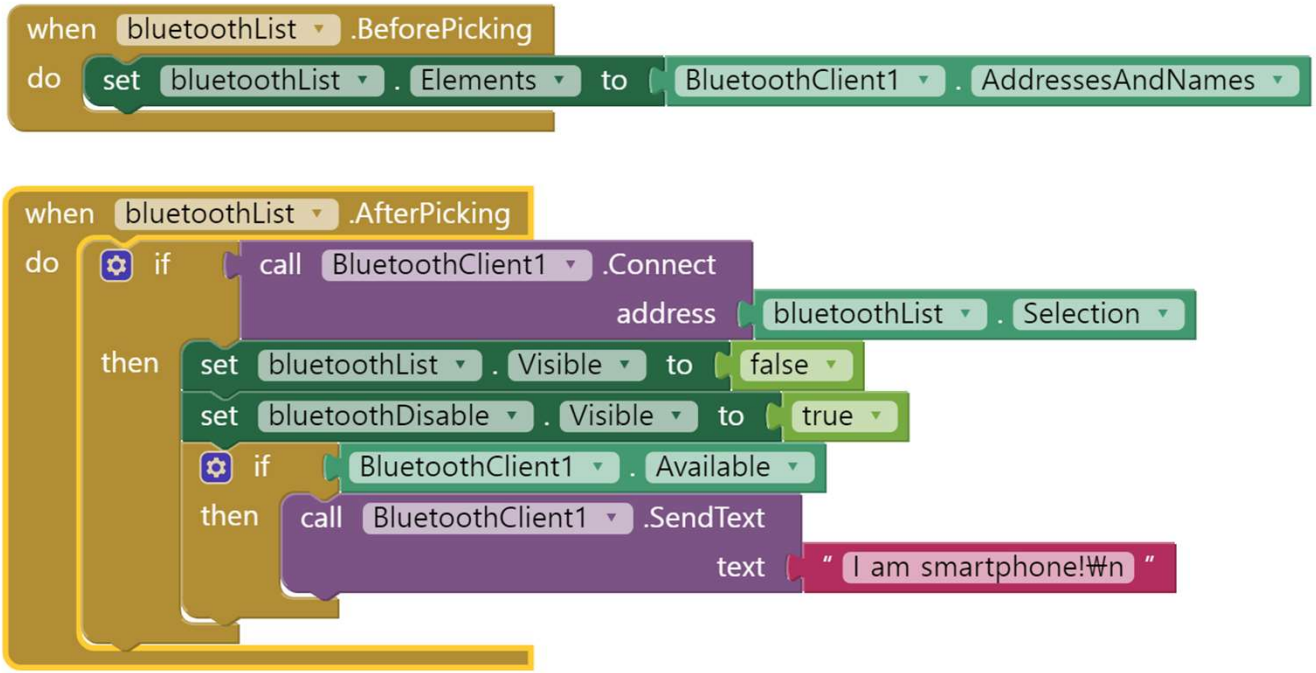
Lab: Simple BT master app

- @ Designer
  - Layout
    - HorizontalArrangement x 1
  - User Interface
    - ListPicker x 1
      - BluetoothList
    - Button x 2
      - BluetoothDisable
      - SendButton
    - Label x 1
      - SlaveText
  - Sensors
    - Clock x 1
      - Clock1 (100 ms)
  - Connectivity
    - BluetoothClient x 1
      - BluetoothClient1



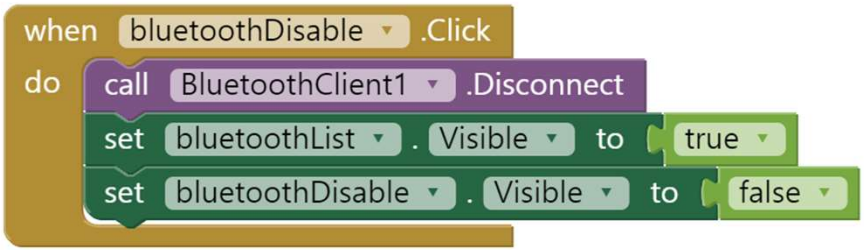
Lab: Simple BT master app

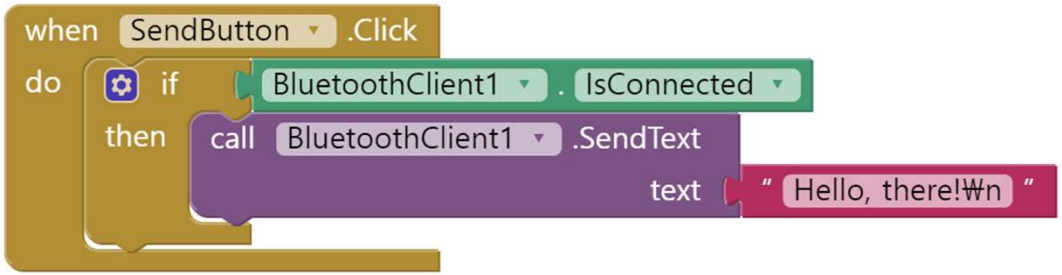
- Screen1 -> HorizontalArrangement -> bluetoothList
- Screen1 -> BluetoothClient1
- Built-in -> Control -> if – then block
- Built-in -> Text -> “I am smartphone₩n”
- Built-in -> Logic -> true or false



Lab: Simple BT master app

- Screen1 -> HorizontalArrangement -> bluetoothDisable
- Screen1 -> BluetoothClient1
- Screen1 -> HorizontalArrangement -> SendButton
- Built-in -> Control -> if – then block
- Built-in -> Text -> "Hello, there₩n"
- Built-in -> Logic -> true or false





Lab: Simple BT master app

- Screen1 -> HorizontalArrangement -> Clock1
- Screen1 -> BluetoothClient1
- Screen1 -> HorizontalArrangement -> SlaveText
- Built-in -> Control -> if – then block
- Built-in -> Variables -> global recvBuf variable
- Built-in -> Text -> "From slaves:₩n", joining two texts
- Built-in -> Logic -> true or false

Assignment: Wireless LED

- Requirements
  - Based on the today examples, write a sketch program as follows.
    - Setup a LED (any color) and BT module on your breadboard
    - Control LED on/off using Bluetooth module
      - Design your own control message. E.g. "on" or "off", "0" or "1", or ....
    - Write a your own smartphone app or use terminal mode of conventional app
  - Write block-type comments in the top of your source code, which includes "your student no., your name, writing date, what you feel about this assignment, etc."
- Results
  - (a source code file) sketch source code (***"sketchfilename.ino"***)
  - (a Arduino board capture file) a demonstration video (max. 1GB file)
    - From setup to execution
  - (optional, a smartphone app file) AppInventor (*.aia)