

APP INVENTOR

# WHAT YOU NEED?

- Android Phone
- MIT Companion App
- Laptop with Internet
- Ideas

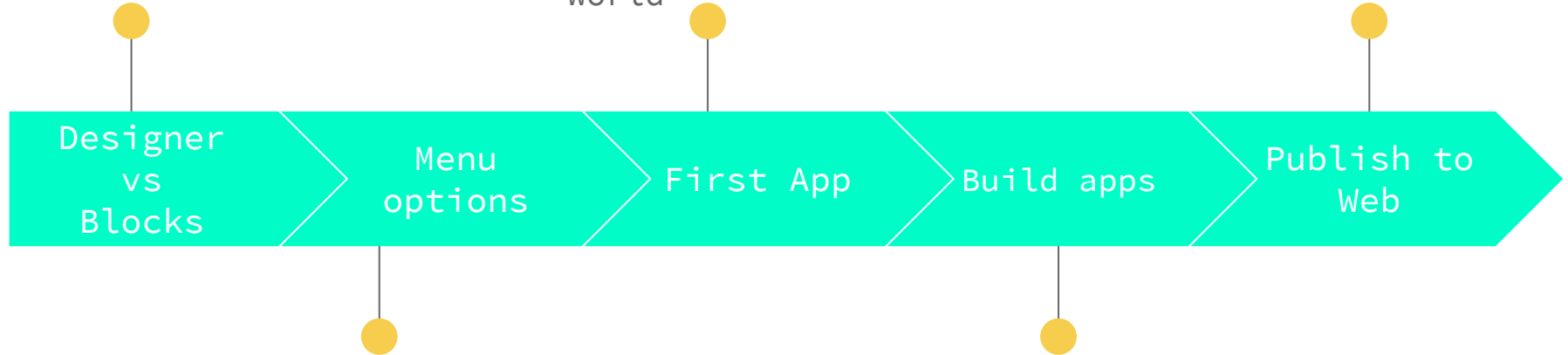
---

WHAT ARE WE GOING  
TO LEARN TODAY?

Describe different options available in designer screen and blocks screen.

A simple hello world app. Click a button and then show text box Hello World

How to publish the app on to Google Play Store using the generated apk



Go through the different options in the menu of App Inventor

Build 2 apps :  
1.Book Scanner  
2.Mobile Piano  
3.Demo  
Temperature(Shield)

MIT App Inventor 2  
Beta

Projects Connect Build Help

My Projects Gallery Guide Report an Issue English gowri.r@gmail.com

Start new project Delete Project Publish to Gallery

My Projects

Name	Date Created	Date Modified	Published
TestProject	May 17, 2017, 10:14:40 AM	Jul 5, 2017, 10:49:23 AM	No
ISBNsearch	Jul 5, 2017, 10:00:01 AM	Jul 5, 2017, 10:32:16 AM	No
Piano_cg	Jul 5, 2017, 10:01:07 AM	Jul 5, 2017, 10:02:42 AM	No
HelloPurr	Jul 5, 2017, 10:01:17 AM	Jul 5, 2017, 10:01:17 AM	No
ballfing	Jun 30, 2017, 10:04:23 AM	Jun 30, 2017, 11:12:17 AM	No
bletemp	Jun 7, 2017, 6:03:13 PM	Jun 30, 2017, 9:49:53 AM	No
Piano	Jun 7, 2017, 11:56:33 AM	Jun 7, 2017, 2:13:48 PM	No

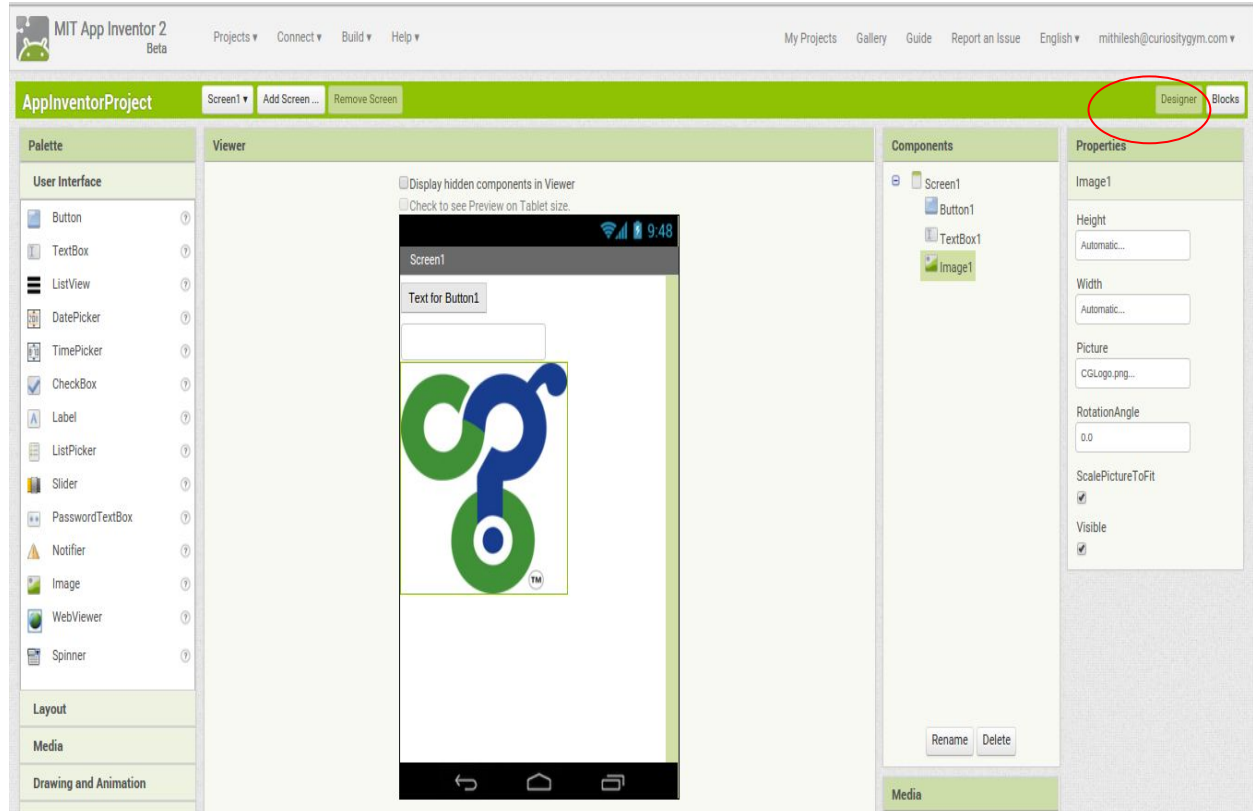
Privacy Policy and Terms of Use

Show all

## START OFF

- Need to have a google account
- Connect to  
“<http://ai2.appinventor.mit.edu/>”
- Start new project
- Delete project
- Publish to gallery:  
publish the code to a common place for all to see

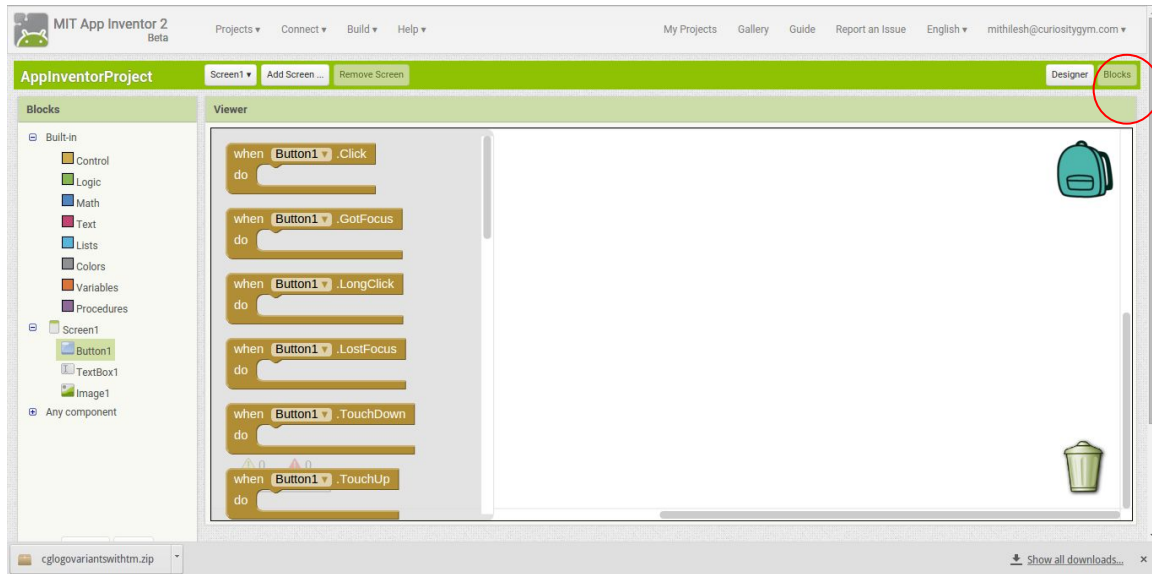
# DESIGNER VS BLOCKS - THE DESIGNER SCREEN



Palette (Designer view - shaded)

- User interface
- Layout
- Media
- Drawing and Sensation
- Sensors
- Social
- Storage
- Connectivity
- LeGO Mindstorms
- Experimental - FirebaseDB on the cloud

# DESIGNER VS BLOCKS - THE BLOCKS SCREEN

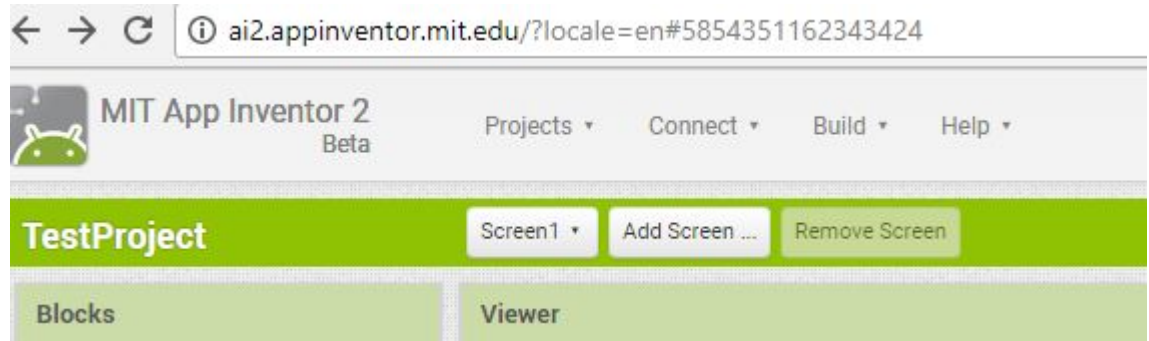


Blocks view (shaded)

- Options
  - Built-in: Programming components
  - Screen: Elements dragged on the screen from the Palette view
  - Any component: Elements dragged on to the screen that are not visible
- Backpack
- Delete

# SCREENS

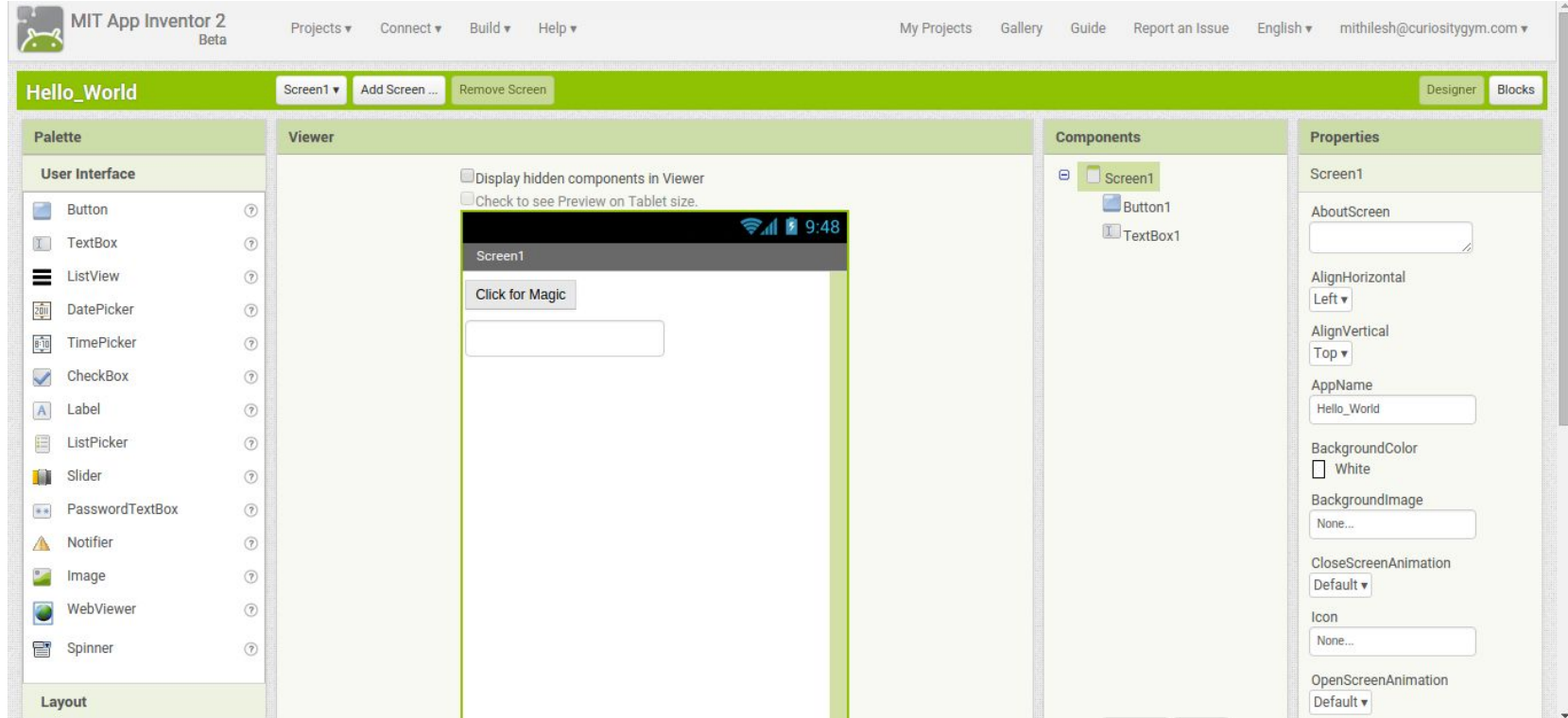
- The default screen is screen1
- Add Screen : if you want to link multiple screens





HELLO WORLD APP

# APP TO DISPLAY HELLO WORLD WHEN A BUTTON IS CLICKED





## Hello\_World

Screen1 ▾

Add Screen ...

Remove Screen

Designer

Blocks

### Blocks

#### Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

- Screen1
  - Button1
  - TextBox1

#### Any component

### Viewer

```
when Button1 .Click  
do set TextBox1 .Text to "Hello World"
```

⚠ 0 🔴 0

Show Warnings



LET'S BUILD MORE  
APPS

# PROBLEM STATEMENTS

1. Build an app which scans a barcode of a book to search for the book on ISBNsearch.org & view the website.
2. Build an app which is a piano. ( White and Black Stripes as buttons, when pressed plays a tone )
3. Use external hardware like CG Shield to build a thermometer app which reports temperature to the mobile.

# PUBLISHED TUTORIALS

- <http://appinventor.mit.edu/explore/ai2/tutorials.html>
- <http://www.appinventor.org/book2>

# GENERAL INSTRUCTIONS

- Do not use the emulator.
  - Very slow and does not work even for a moderately complex app.
  - *BarCode Scanner App is needed on phones that don't have it. Go to play store and install it*
- Check if **download unknown apps** is checked. Then app won't work
- Download MIT AI2 companion from Play Store
- You have internet?
  - You have a camera?
    - “Build->Provide QR code for app” option
  - Don't have camera?
    - “Connect->AI2 companion” to get the QR code and manually enter it on the AI2 appinventor application on the phone
- No internet?
  - “Build->Save .apk to my computer” and transfer it onto the phone using a physical connector cable

PIANO APP



# PIANO APP

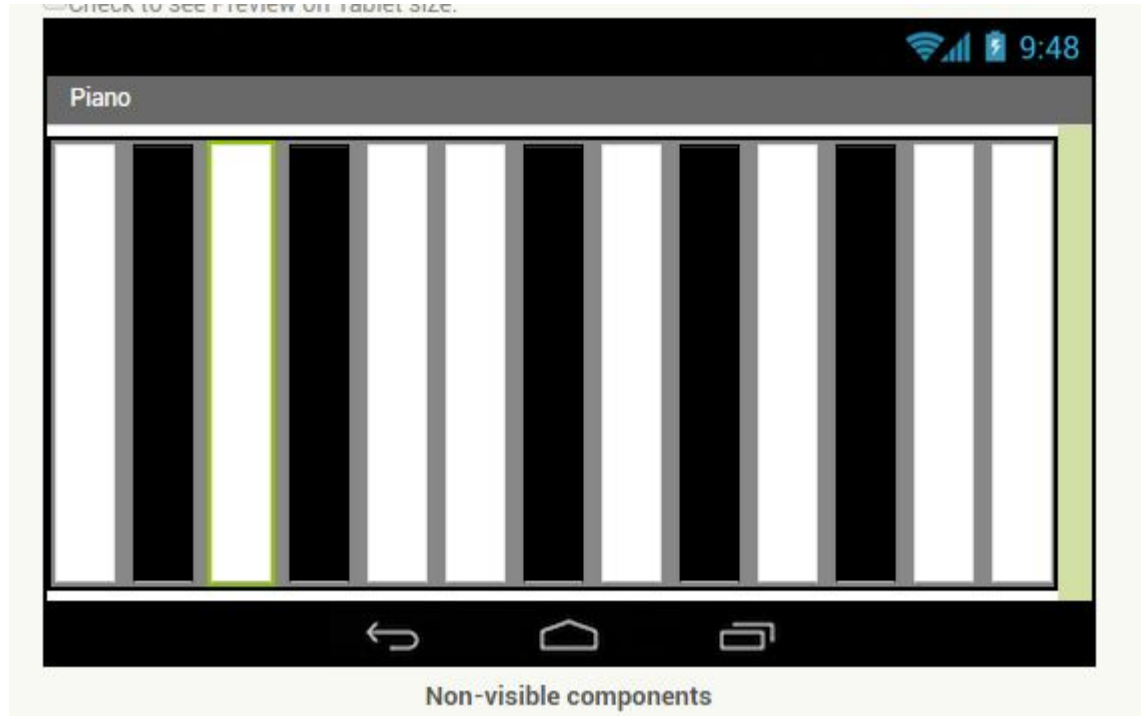
- Download the sounds from  
<https://freesound.org/people/pinkyfinger/packs/4409/>
- Need to register
- Uploaded on github  
<https://github.com/sahirvsahirv/CSTeachingMaterial/tree/master/AppInventor/PianoApp>
- Read up on  
<http://www.howmusicworks.org/107/Sound-and-Music/Octaves>  
and the next 5 pages

# PIANO APP

- Need 12 keys for one octave
- Add 12 buttons in the White and Black pattern in the previous slide
- Starts with C

1	2	3	4	5	6	7	8	9	10	11	12
W		W		W	W		W		W		W
C	C# (Db)	D	D# (Eb)	E	F	F# (Gb)	G	G# (Ab)	A	A# (Bb)	B

# PIANO APP



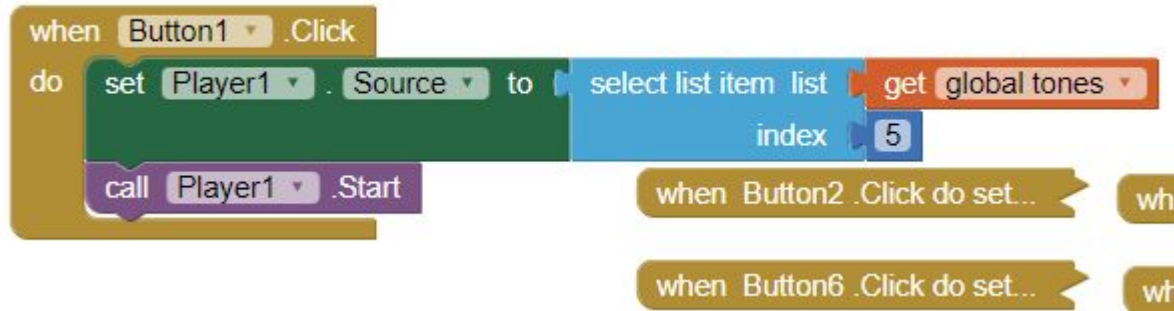
# PIANO APP

- In the designer->Media upload the .wav files
- Create a list of the media files with the same names of the files



# PIANO APP

- Add this code to all the 12 buttons – the indexing of the lists starts from 1 and the corresponding .wav files to be used is in the [slide 18](#). The .wav files are automatically picked up from the source



GET TEMPERATURE  
THROUGH BLUETOOTH

# COMPONENTS NEEDED

- HC-05 - per head
- Arduino with or without Curiosity Gym Idiot Ware shield - per head
- USB to TTL converter handy for AT mode - just one pre configured to reduce issues
- Laptop for AppInventor application
- LM35 or DHT11 sensor to read the temperature (LM35 already on the Idiotware shield) - per head

# HC-05

- A cheap master slave bluetooth device
- Uses Standard serial data and transmits it via bluetooth
  - Serial SPP (Serial Port Protocol)
    - **This point is important to know how to read data from Android**
- Can be in master or slave setup. By default it is in slave mode
- If we need to change it
  - Use Arduino and get the Key pin to high and reset HC-05 at the same time
  - Use USB-TTL converter - press reset and power key and 5V at the same time. When it is in configurable mode, the light flashes slowly and in connection mode, it flashes rapidly
  - Used to change the name of the HC-05 module when in Arduino-Arduino connections, many are connected together and
  - To get the baud rate too if it is changed, default is 38400



# BLUETOOTH WITH ANDROID'S BLUETOOTH

- Bluetooth protocol supports multiple devices, one master and many slaves speaking to each other
- Android's implementation of the Bluetooth stack allows only one Master and one Slave
- This is done by **pairing - Inquiry part of the protocol**
- Pairing needs an authentication code, since we are pairing the HC-05 module on the Arduino - we give "1234", the default given in the datasheet
- Go to Settings on the phone and enable Bluetooth - The Android device searches for nearby bluetooth devices. Acts as the Master here **since it initiates a connection**
- We can make Android's Bluetooth as the Slave and Arduino's Bluetooth as Master using Android code and

# REVELATIONS

- Since it is invisible and transparent (SPP), it is like connecting two wires to RX and TX of the Arduino
- Instead of a Android phone, you pair HC-05 with your laptop, you will see it showing up using 2 COM ports on the device manager on a windows machine.
- Use BlueTerm or Termite (any serial port reading software) to connect to 38400 baud rate (default - that HC-05 uses) to see what data is being sent by Arduino
- We are going to use a code in AppInventor to read the same

# PROTOCOL

- What should we code in the Android application?
- Master(Android phone and also Bluetooth Client)
- Slave(HC-05 and also Bluetooth Server)
- Slave needs to [PageScan] and Master needs to [Page] according to the protocol to complete the connection

# WIRE CONNECTIONS ON ARDUINO

- HC-05 (Rx) to Tx(on Arduino) and HC-05(Tx) to RX(on Arduino) - directly connecting to Arduino
- Using the shield, just align the 5V on the shield and 5V on the HC-05
- On the shield the Key pin is not soldered and hence can go to AT - mode using the shield
- To connect HC-05 to the USB to TTL converter (KEY has to be pulled HIGH) - cross connection here too (TXtoRX and RXtoTX)
- The HC-05 module that we use is on the breakout board and has a voltage divider to power down the Arduino 5V power supply, we need not bother about it

# CODE ON ARDUINO SIDE

- Download the software serial example and reuse it for testing basic bluetooth communication
- To use LM35 using the shield put the jumper on and read A0 analog pin on Arduino, else power to 5V, A0 and ground for direct connection with Arduino
- To read DHT11, use one of the digital pins, 5V and ground
- Using the shield need not bother about the jumper since it uses digital pins

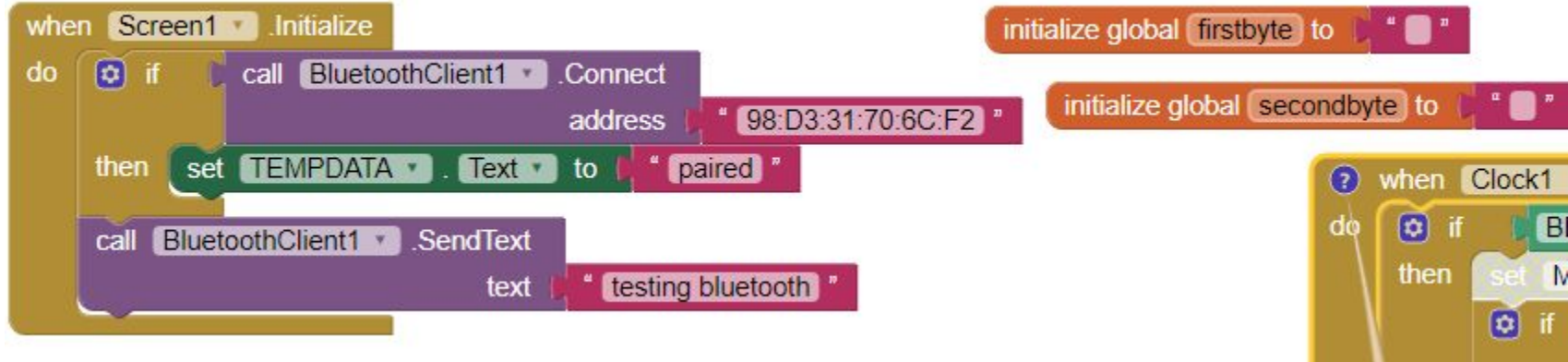
# CODE ON ARDUINO

- If the LED display is used and D1 is used for ON and OFF using the shield, Serial.begin should not be there, since D0 and D1 are Rx and TX for serial connection
- The software serial (Bluetooth wireless serial ports should use some other digital pins other than D0 and D1)
- While using the shield, the Bluetooth's TX is soldered to D10 on the Arduino (which will act as RX)
- And HC-05's RX is soldered to D9 on the Arduino (which will act as TX)
- For DHT11 use the AdaFruit DHT11 library and import it using "Manage Libraries"
- For LM35 voltage to celsius conversion formula to be used

# CODE

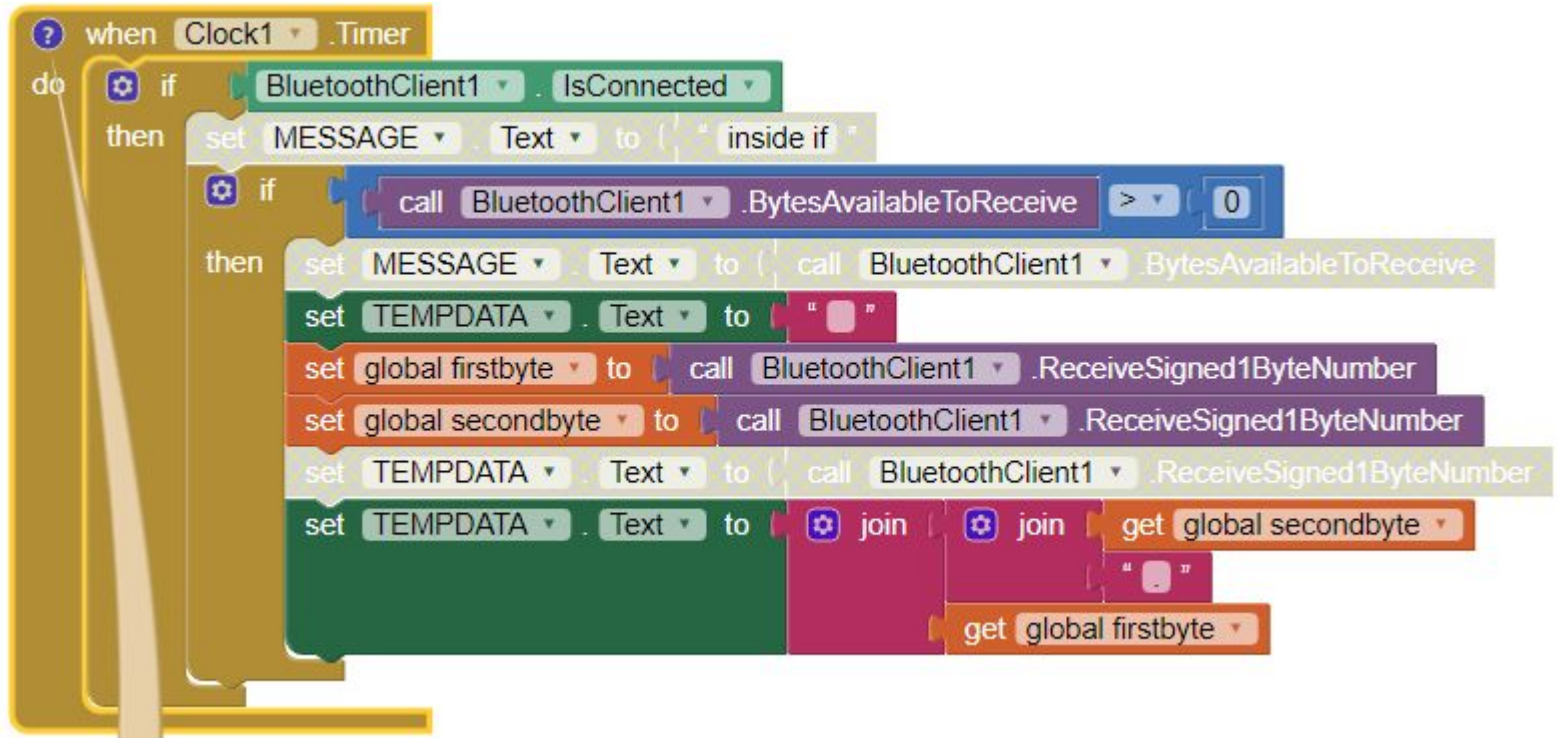
- [https://github.com/sahirvsahirv/CSTeachingMaterial/blob/master/AppInventor/Bluetooth/bluetooth\\_android\\_send\\_receive.ino](https://github.com/sahirvsahirv/CSTeachingMaterial/blob/master/AppInventor/Bluetooth/bluetooth_android_send_receive.ino)
- Base is the software serial example to read on bluetooth and write on to it
- The other void loop() below is to receive ON/OFF signals from Android to switch ON/OFF and LED

# ANDROID CODE





# ANDROID CODE



# ANDROID CODE

- The android code does not read every single temperature reading that is being sent by Arduino
- It is lossy
- The beginning communication to reply to the “PageScan” sent by Arduino. It is called the “Page” (Sending “Testing Bluetooth”)
- We read only one byte at a time (temperature fits in less than a byte and data is not split) since Bluetooth uses SPP (explained in the earlier slides)
- And since the **Highest Order Byte is sent first**, we swap it and read
- Since it is sort of synchronized with the clock timings

# ANDROID CODE - ON/OFF LED

when OFF .Click  
do call BluetoothClient1 .Send1ByteNumber  
number 0

when ON .Click  
do call BluetoothClient1 .Send1ByteNumber  
number 1

when DISCONNECT .Click  
do call BluetoothClient1 .Disconnect

send and r  
temperatur

User tapped and released the button.

"Bluetooth - arduino sending 0 and 1 to switch on..."

0 0  
how Warnings

WEB SERVICE APP

# COMPONENTS TO INTERACT WITH THE WEB

## (CHAPTER 24 BOOK 2)

- User Interface->WebView:
  - UI element, dragged onto the screen. Connect to the webpage and the web page contents are shown in the user interface.
  - Example: [https://en.wikipedia.org/wiki/Prime\\_number](https://en.wikipedia.org/wiki/Prime_number)
- Connectivity->Web:
  - This is to connect to a web service API
  - Any generic URL will not work
  - Web service API: It is an interface for writing your own software code that can connect to the URL that does not give content that read easily visually, but content that a computer can read through code easily (xml/yaml/json/csv)
  - Example:
    - <http://isbndb.com/api/v2/json/D6ONXMNB/book/9780131988132>
    - <http://samples.openweathermap.org/data/2.5/forecast?id=524901&appid=b1b15e88fa797225412429c1c50c122a1>

# COMPONENTS TO INTERACT WITH THE WEB

- TinyWebDB:
  - The phone can interact with an application for Multi-Player Games
  - Exposes GetValue and SetValue
  - Complex parsing cannot be done in AppInventor code
  - Such applications are written as a web URL hosted somewhere and that web URL exposes GetValue and SetValue functions on the web
  - The complexity is hidden in the web URL code and only tag-value pairs are received and the data is read
  - Storing the data is done by the webservice
  - From a huge JSON text we need just the “author” tag and the “author-name” for a specific book(ISBN), we retrieve just that.

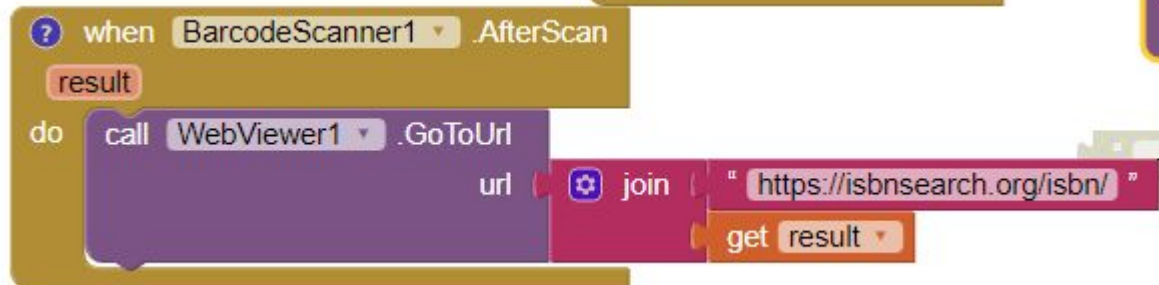
# DATA RETURNED: JSON/XML/YAML

Sample JSON file:

```
{ "markers": [  
  {  
    "point": new GLatLng(40.266044,-74.718479),  
    "homeTeam": "Lawrence Library",  
  },  
  {  
    "point": new GLatLng(40.211600,-74.695702),  
    "homeTeam": "Hamilton Library",  
  },  
]
```

# WEB VIEWER

- This link is <https://isbnsearch.org>
- They have introduced a captcha in it
- You see the captcha page
- A workaround could be:
  - On a button click, show the web page and indicate that the user resolves the captcha.
  - And next ask the user to search for the required ISBN number manually





# WEB

- Web1.SaveResponse should be set to false so that we the “GotText” event
- The URL can be set from the designer too to any RESTful API, even to one hosted by you
- We shall host one soon
- <http://192.168.1.7/getvalue.php>
- Try <http://samples.openweathermap.org/data/2.5/forecast?id=524901&appid=b1b15e88fa797225412429c1c50c122a1>:Complex Scraping

```
when Screen1.Initialize
do
  set Web1.Url to "http://isbndb.com/api/v2/json/D6ONXMNB/book/9780..."
  set Web1.SaveResponse to false
  call Web1.Get
```

```
when Web1.GotText
  url responseCode responseType responseContent
do
  set Button1.Text to get responseCode
  set Label1.Text to get responseContent
  set TextBox2.Text to get responseType
```

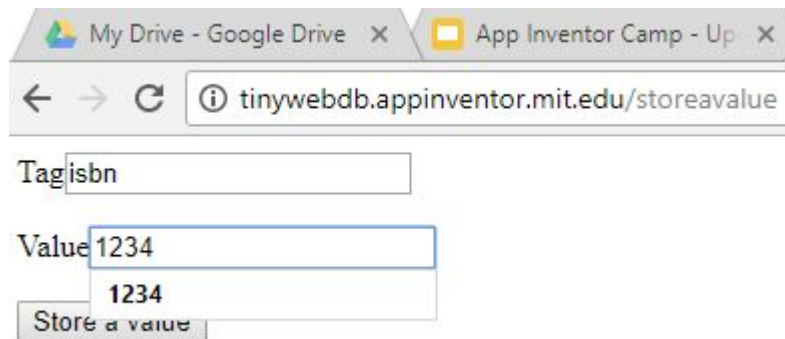
⚠ 15 ❌ 13  
Show Warnings

```
"http://samples.openweathermap.org/data/2.5/forec..."
"http://finance.yahoo.com/d/quotes.csv?f=1&s=GOOG"

? when Web1.GotFile
  url responseCode responseType fileName
do
  set Button1.Text to get responseCode
  set Label1.Text to get fileName
  set TextBox2.Text to get responseType
```

# TINYWEBDB COMPONENT

- Where do you host your web service
  - On the cloud  
<http://tinywebdb.appspot.com>
  - Only 250 entries allowed and individual data limited to 500 characters
  - JSON requests are allowed
  - Access  
<http://tinywebdb.appspot.com/storeavalue> and give tag and value



My Drive - Google Drive x App Inventor Camp - Up x

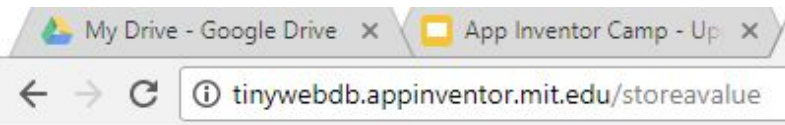
← → ↻ ⓘ tinywebdb.appspot.com/storeavalue

Tag isbn

Value 1234

1234

Store a value



My Drive - Google Drive x App Inventor Camp - Up x

← → ↻ ⓘ tinywebdb.appspot.com/storeavalue

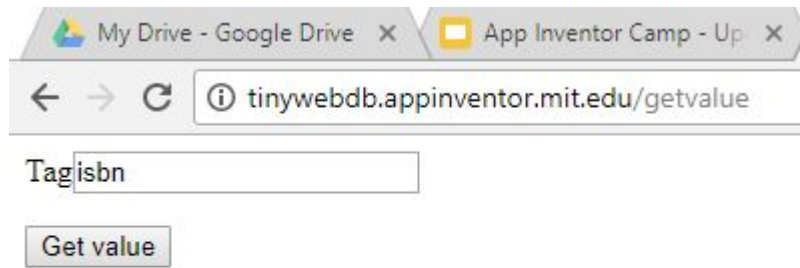
*The server will send this to the component:*

`["STORED", "isbn", "1234"]`

[Return to TinyWebDB Main Page](#)

# CLOUD ACCESS

- <http://tinywebdb.appinventor.mit.edu/getvalue>  
and give the tag  
("isbn") and receive  
the value "1234" as  
JSON

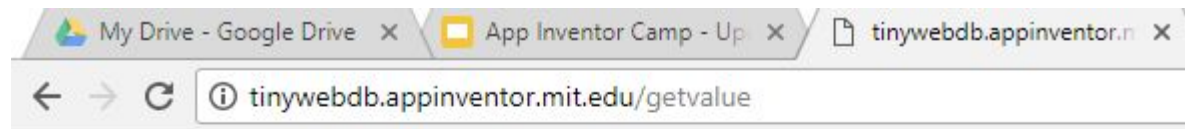


My Drive - Google Drive x App Inventor Camp - Up x

← → ↻ ⓘ tinywebdb.appinventor.mit.edu/getvalue

Tag isbn

Get value



My Drive - Google Drive x App Inventor Camp - Up x tinywebdb.appinventor.n x

← → ↻ ⓘ tinywebdb.appinventor.mit.edu/getvalue

*The server will send this to the component:*

`["VALUE", "isbn", "1234"]`

[Return to TinyWebDB Main Page](#)

# CODE

```
when TinyWebDB3 .GotValue
  tagFromWebDB valueFromWebDB
do set TextBox1 . Text to get valueFromWebDB
```

```
when Screen1 .Initialize
do call TinyWebDB3 .StoreValue
  tag "isbn"
  valueToStore "1234"
```

```
when TinyWebDB3 .ValueStored
do call TinyWebDB3 .GetValue
  tag "isbn"
```

# TINYWEBDB - GOOGLE APP ENGINE

- Need Card details
- It is a cloud platform that has all installed so that only the code can be focussed on to make web and mobile apps
- Read this: <https://cloud.google.com/appengine/downloads>
- To emulate on the desktop environment  
<http://www-personal.umich.edu/~csev/books/gae/handouts/AppEngine-Install-Windows.pdf>
- The cloud platform requires a credit card number for logging in
- Steps to host it on the cloud -  
<http://appinventor.mit.edu/explore/ai2/custom-tinywebdb.html>

# TINY WEB DB

- Host own web service on a local web server
- WAMP is a PaaS has the web server and database and PHP run time installed
- WSGI is for running python apps on the PaaS
- Install WAMP for windows from <https://sourceforge.net/projects/wampserver/>
- Remember to shut the service “Web Deployment Agent Service” for WAMP to run
- Install AMPPS <http://www.ampps.com/> for a cross platform one that has WSGI installed
- We shall develop a PHP webservice
- Look into the log files for debugging

# PHP WEB SERVICE

- In the apache configuration file at  
"C:\wamp64\bin\apache\apache2.4.23\conf\httpd.conf"
- uncomment "LoadModule rewrite\_module modules/mod\_rewrite.so"
- I added it below all the lines under LoadModule php5\_module  
"\${INSTALL\_DIR}/bin/php/php5.6.25/php5apache2\_4.dll"
- Need cURL to write code to accessing remote files. HTTP requests are made using cURL
  - C:\wamp64\bin\php\php7.0.10\php.ini
  - Uncomment "extension=php\_curl.dll" (Remove the ';')
  - "allow\_url\_fopen = On" for HTTP requests without curl
- Add the following lines. Give a sub directory - "RewriteBase /isbnscraper" for accessing it as /isbscraper/getvalue
  - #For the getvalue and setvalue webservice with appInventor
  - <IfModule mod\_rewrite>
  - RewriteEngine On
  - RewriteBase /
  - RewriteRule ^getvalue\$ **getvalue.php**
  - RewriteRule ^setvalue\$ **storeavalue.php**
  - </IfModule>

# CUSTOM WEB SERVICE

- Create two files “`getvalue.php`” and “`storeavalue.php`” in the root folder of the WAMP server. In my case, `C:\wamp64\www`
- Use notepad or Webmatrix as the editor
- Drag two TinyWebDB components on the Designer
- Set the URLs to
  - <http://192.168.1.7/storeavalue.php> and
  - <http://192.168.1.7/getvalue.php>
- Write the webservice code
- Write AppInventor code
- <http://isbndb.com/account/logincreate> - to get the KEY for using the ISBN API
- **"D60NXMNB"** - My key



# JSON CONTENT

- <http://isbndb.com/api/v2/json/D60NXMNB/book/9780131988132>

# CODE ALGORITHM

- AppInventor calls “storeavalue.php” and the “tag” and “value” sent are extracted in the code
- The isbn value sent as value is important to us and we store in a file on the same folder “isbnvalue.txt”
- The value is sent with quotes, trim it
- The “getvalue.php” is called from AppInventor and the isbn value is read from the “isbnvalue.txt”
- Form the url “<http://isbndb.com/api/v2/json/D60NXMNB/book/9780131988132>” by appending the isbn received
- Receive the JSON data using cURL
- Scraping: Extract only the author’s name since we intend to show only that
- Form the JSON text in the format required, ["VALUE","author","'Author’s name'"]
- Last step is **very important**. If not in the same format, AppInventor would not read the tag(“author”) and value (“author’s name”) properly
- Display it on the UI
- Incorporate the barcode scanner on phones that have it for scanning and retrieving the barcode automatically

# FURTHER STEPS

- Use AMPPS to create a python web service
- Store the isbn number in a database instead of a text file
- Use your parent's credit card number to use Google's or Amazon's cloud service to store the isbn number
- Use google app engine SDK to emulate the Google cloud environment

# IN SHORT

APP INVENTOR  
CODE  
Scans barcode  
Displays author's  
name

ISBN



Author's  
name

Web service (GET and SET)  
hosted on a any server over  
HTTP  
(Scraper)

ISBN no



Whole  
JSON text

ISBN Web API

PUBLISH TO PLAY  
STORE

# PUBLISH THE APP TO GOOGLE PLAY STORE

1. Extract the APK from App Inventor
2. Login to Google Dev Account
3. Pay the Play Store Charges
4. Publish your app.