Τεχνολογία και Προγραμματισμός Κινητών Συσκευών

10

Διαχείριση Bluetooth

Ιωάννης Έλληνας

Επικοινωνία με Bluetooth

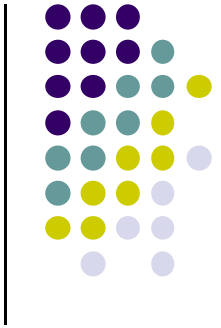
Οι συσκευές που διαθέτουν το ασύρματο πρωτόκολλο Bluetooth έχουν τις ακόλουθες δυνατότητες:

- Ανίχνευση συσκευών που διαθέτουν BT.
- Αναζήτηση στον τοπικό προσαρμογέα (BT adapter) των συσκευών που έχουν συνδεθεί (paired ή bonded).
- Δημιουργία καναλιών RFCOMM.
- Ανταλλαγή δεδομένων με άλλες συσκευές.
- Διαχείριση πολλαπλών συνδέσεων.
- Το RFCOMM (Radio Frequency COMMunication) είναι ένα πρωτόκολλο μετάδοσης δεδομένων το οποίο προσομοιώνει τη λειτουργία του RS232 για τις συσκευές BT.

Κλάσεις και διασυνδέσεις των APIs που διαθέτει μια κινητή συσκευή για το Bluetooth:

- **BluetoothAdapter** – **Προσαρμογέας για όλους τους χειρισμούς BT**. Με αυτόν μπορούν να ανιχνευθούν άλλες συσκευές BT, να αναζητηθούν ήδη συνδεδεμένες συσκευές, να αρχικοποιηθεί μια συσκευή BT με την MAC διεύθυνσή της (Media Access Control Address) και να δημιουργηθεί ένα κανάλι επικοινωνίας με άλλες συσκευές (BluetoothServerSocket).
- **BluetoothDevice – Αντιπροσωπεύει μια απομακρυσμένη συσκευή BT**. Χρησιμοποιείται για αίτηση σύνδεσης με αυτήν μέσα από ένα κανάλι (BluetoothSocket) ή για την αναζήτηση πληροφοριών για τη συσκευή (όνομα, διεύθυνση, κατάσταση σύνδεσης).
- **BluetoothSocket – Αντιπροσωπεύει τη διασύνδεση με ένα κανάλι επικοινωνίας (Bluetooth socket)**. Είναι το σημείο σύνδεσης με το κανάλι για την ανταλλαγή δεδομένων μέσω των *InputStream* και *OutputStream*.
- **BluetoothServerSocket – Αντιπροσωπεύει ένα ανοικτό κανάλι εξυπηρετητή που δέχεται αιτήσεις εξυπηρέτησης**. Για την επικοινωνία δύο συσκευών, πρέπει η μία να δημιουργήσει ένα server socket και όταν μια άλλη συσκευή ζητήσει σύνδεση, ο εξυπηρετητής θα δημιουργήσει ένα Bluetoothsocket μόλις η σύνδεση γίνει αποδεκτή.
- **BluetoothClass** – Περιγράφει τα χαρακτηριστικά και τις δυνατότητες μιας συσκευής BT.
- **BluetoothProfile** – Προδιαγραφές για την επικοινωνία δύο συσκευών (π.χ. το προφίλ επικοινωνίας συσκευών hands-free).

Χρήση Bluetooth



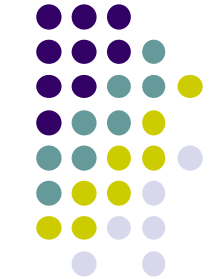
- Η χρήση του Bluetooth από την κινητή συσκευή γίνεται αφού εξασφαλιστεί άδεια χρήσης από το λειτουργικό. Έτσι στο αρχείο μανιφέστου πρέπει να τοποθετηθεί η παρακάτω ιδιότητα:

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

- Για την αναζήτηση άλλων συσκευών BT ή για τη διαχείριση των δυνατοτήτων BT πρέπει να ζητηθεί άδεια διαχειριστή:

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

Ενεργοποίηση Bluetooth



Καλείται ο προσαρμογέας Bluetooth με τη μέθοδο *getDefaultAdapter()*. Ο προσαρμογέας είναι μοναδικός για κάθε συσκευή και οι εφαρμογές επικοινωνούν κάθε φορά με αυτόν.

```
BluetoothAdapter mBluetoothAdapter=BluetoothAdapter.getDefaultAdapter();
if (mBluetoothAdapter == null) {
  // Device does not support Bluetooth
}
```

Για να ελέγξουμε αν το Bluetooth είναι ενεργοποιημένο, καλούμε τη μέθοδο *isEnabled()*. Για να το ενεργοποιήσουμε, αν η μέθοδος επιστρέψει false, χρησιμοποιούμε τη μέθοδο *startActivityForResult()* με την πρόθεση ACTION_REQUEST_ENABLE. Έτσι θα εκπεμφθεί προς το σύστημα μια πρόθεση ενεργοποίησης του Bluetooth.

```
if (!mBluetoothAdapter.isEnabled()) {
  Intent enableBtIntent = new Intent(BluetoothAdapter. ACTION_REQUEST_ENABLE)
  startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
}
```

requestCode.
Η onActivityResult() επιστρέφει το resultCode που είναι RESULT_OK ή RESULT_CANCELED

Αναζήτηση Συσκευών BT

Μέσω του προσαρμογέα *BluetoothAdapter* βρίσκονται συσκευές BT ή δίνεται λίστα των συσκευών που έχουν ήδη συνδεθεί (bonded devices).

- Η συσκευή αποστέλλει μια αίτηση ανίχνευσης και οι συσκευές που βρίσκονται στο πεδίο της και έχουν ενεργοποιημένη την αντίστοιχη λειτουργία αποστέλλουν το όνομα και τη διεύθυνση MAC.
- Το σύστημα παρουσιάζει τις συσκευές στον χρήστη, ο οποίος στη συνέχεια επιλέγει με ποια συσκευή θέλει να συνδεθεί, οπότε δημιουργείται ένα ζεύγος συσκευών (paired devices) και αφού τα στοιχεία της απομακρυσμένης συσκευής αποθηκευτούν μπορεί να ξεκινήσει η επικοινωνία τους.
- Αν η διεύθυνση MAC μιας απομακρυσμένης συσκευής είναι γνωστή, τότε μπορεί να γίνει σύνδεση μαζί της χωρίς να προηγηθεί η διαδικασία αναζήτησης συσκευών.
- Να σημειωθεί ότι οι συσκευές πρώτα γίνονται ζεύγος (paired), το οποίο σημαίνει ότι γνωρίζουν η μία την ύπαρξη της άλλης και μπορούν να αποκαταστήσουν μια σύνδεση μεταξύ τους, και μετά μοιράζονται ένα κανάλι σύνδεσης RFCOMM για ανταλλαγή δεδομένων.

Αναζήτηση Συζευγμένων Συσκευών

Πριν την αναζήτηση συσκευών, ζητείται λίστα των συζευγμένων συσκευών με τη μέθοδο *getBondedDevices()*, η οποία επιστρέφει τις συσκευές που είναι συνδεδεμένες.

Το αντικείμενο *BluetoothDevice* χρειάζεται τη διεύθυνση MAC της απομακρυσμένης συσκευής, η οποία μπορεί να ανασυρθεί από τον πίνακα αποθήκευσης, για να πραγματοποιήσει μια σύνδεση.

```
Set<BluetoothDevice> pairedDevices=mBluetoothAdapter.getBondedDevices();
// If there are paired devices
if (pairedDevices.size() > 0) {
  // Loop through paired devices
  for (BluetoothDevice device : pairedDevices) {
    // Add name and address to an array adapter to show in a ListView
    mArrayAdapter.add(device.getName() + "\n" + device.getAddress());
  }
}
```

Διαδικασία αναζήτησης

- Έναρξη αναζήτησης συσκευών με τη μέθοδο ***startDiscovery()***. Η μέθοδος επιστρέφει μια boolean τιμή ότι άρχισε η σάρωση η οποία επαναλαμβάνεται κάθε 12 δευτερόλεπτα.
- Δημιουργία δέκτη μηνυμάτων εκπομπής για την πρόθεση **ACTION_FOUND** για να λαμβάνει πληροφορίες για κάθε συσκευή που ανιχνεύεται. Η ανίχνευση μιας συσκευής συνοδεύεται με τη μεταφορά των πεδίων EXTRA_DEVICE και EXTRA_CLASS που εμπεριέχουν μια BluetoothDevice και μια BluetoothClass.

Εγγραφή Δέκτη

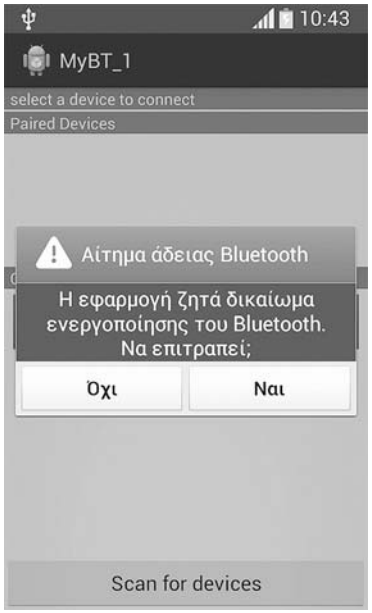
```
// Register the BroadcastReceiver
IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
registerReceiver(mReceiver, filter);        // unregister during onDestroy
```

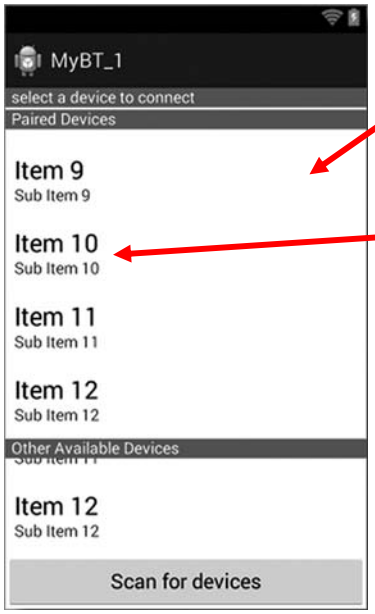
Λειτουργία Δέκτη

```
// Create a BroadcastReceiver for ACTION_FOUND
private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
  public void onReceive(Context context, Intent intent) {
    String action = intent.getAction();
    // When discovery finds a device
    if (BluetoothDevice.ACTION_FOUND.equals(action)) {
      // Get the BluetoothDevice object from the Intent
      BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
     // Add name and address to an array adapter to show in a ListView
     mArrayAdapter.add(device.getName() + "\n" + device.getAddress());
    }
  }
};
```

Παράδειγμα αναζήτησης (MyBT_1)

Η εφαρμογή ελέγχει αν η συσκευή Android έχει ενεργοποιημένη τη λειτουργία Bluetooth και ζητά άδεια ενεργοποίησης. Στη συνέχεια καταγράφει τις συσκευές με τις οποίες έχει συνδεθεί στο παρελθόν (bonded devices), ανιχνεύει τις BT συσκευές που υπάρχουν κοντά της και δημιουργεί λίστα για κάθε κατηγορία αυτών των συσκευών.



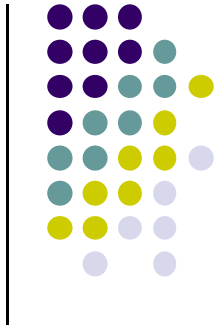


activity_main.xml

device_name.xml

Αρχείο μανιφέστου

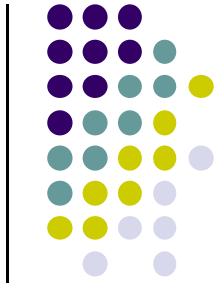
```
<manifest
  ……
  <uses-sdk
    android:minSdkVersion="10"
    android:targetSdkVersion="18" />
  <uses-permission android:name="android.permission. BLUETOOTH"/>
  <uses-permission android:name="android.permission. BLUETOOTH_ADMIN"/>
  ……
</manifest>
```

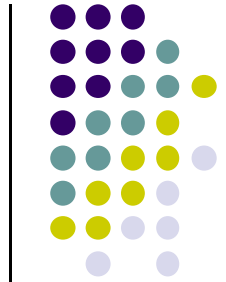


```
package com.example.mybt_1;
import ……
public class MainActivity extends Activity {
  TextView txt1, txt2, txt3;

  private static final int REQUEST_ENABLE_BT = 1;
  private BluetoothAdapter mBluetoothAdapter;
  private ArrayAdapter<String> mPairedDevicesArrayAdapter;
  private ArrayAdapter<String> mNewDevicesArrayAdapter;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    txt1 = (TextView) findViewById(R.id.textView1);
    txt2 = (TextView) findViewById(R.id.textView2);
    txt3 = (TextView) findViewById(R.id.textView3);

    // New array adapters for paired devices and for new devices
    mPairedDevicesArrayAdapter = new ArrayAdapter<String>(this, R.layout.device_name);
    mNewDevicesArrayAdapter = new ArrayAdapter<String>(this, R.layout.device_name);

    // Set up ListView for paired devices and add listener
    ListView pairedListView = (ListView) findViewById(R.id.paired_devices);
    pairedListView.setAdapter(mPairedDevicesArrayAdapter);
    pairedListView.setOnItemClickListener(mDeviceClickListener);

    // Set up ListView for new devices and add listener
    ListView  newDevicesListView = (ListView) findViewById(R.id.new_devices);
    newDevicesListView.setAdapter(mNewDevicesArrayAdapter);
    newDevicesListView.setOnItemClickListener(mDeviceClickListener);
```

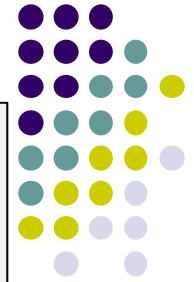
```
  // Register for broadcasts when a device is discovered
  IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
  this.registerReceiver(mReceiver, filter);

  // Register for broadcasts when discovery has finished
  filter = new IntentFilter(BluetoothAdapter. ACTION_DISCOVERY_FINISHED);
  this.registerReceiver(mReceiver, filter);

  // Get the local Bluetooth adapter
  mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
  // If adapter is null, then Bluetooth is not supported
  if (mBluetoothAdapter == null) {
    Toast.makeText(getBaseContext(), "Bluetooth not supported...", Toast.LENGTH_LONG).show();
    finish();
    return;
  }
}

@Override
protected void onStart() {
  super.onStart();
  // If BT is OFF, request to enable it during onActivityResult
  if (!mBluetoothAdapter.isEnabled()) {
    Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
  }
  else {
    discoverBT();
  }
}
```

```
@Override
protected void onResume() {
  super.onResume();
}

@Override
protected void onDestroy() {
  super.onDestroy();
  // Cancel discovery if it is on
  if (mBluetoothAdapter != null) {
    mBluetoothAdapter.cancelDiscovery();
  }
  // Unregister broadcast receiver
  this.unregisterReceiver(mReceiver);
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent  data) {
  switch (requestCode) {
  case REQUEST_ENABLE_BT:
    // The request to enable Bluetooth returns OK
    if (resultCode == Activity.RESULT_OK) {
      // Bluetooth is now enabled, proceed to discover BT devices
      discoverBT();
    } else {
        // Bluetooth was not enabled or there is an error
        Toast.makeText(this, "Bluetooth not enabled...", Toast.LENGTH_SHORT).show();
        finish();
      }
   }
}
```

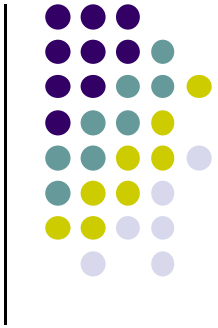




```
public void discoverBT() {
  // Get currently paired devices
  Set<BluetoothDevice> pairedDevices = mBluetoothAdapter.getBondedDevices();
  // If there are paired devices, add them to the ArrayAdapter
  if (pairedDevices.size() > 0) {
    for (BluetoothDevice device : pairedDevices) {
      mPairedDevicesArrayAdapter.add(device.getName() + "\n" + device.getAddress());
    }
  } else {
      String noDevices =  getResources().getText(R.string. none_paired).toString();
      mPairedDevicesArrayAdapter.add(noDevices);
  }
}

public void startScan(View v) {
  v.setVisibility(View.GONE);
  // Stop discovering if already is running
  if (mBluetoothAdapter.isDiscovering()) {
    mBluetoothAdapter.cancelDiscovery();
  }
  txt1.setText(R.string.scanning);
  txt1.setBackgroundResource(R.color.scan_background);
  // Request discover from BluetoothAdapter
  mBluetoothAdapter.startDiscovery();
}
```



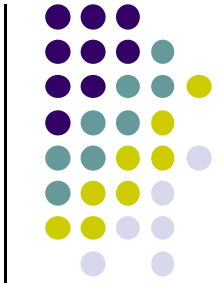
```
// Listener for all devices in the ListViews
 private OnItemClickListener mDeviceClickListener = new OnItemClickListener() {
   public void onItemClick(AdapterView<?> av, View v, int arg2, long arg3) {
     // Cancel discovery in order to connect
     mBluetoothAdapter.cancelDiscovery();
     // Get device MAC address (the last 17 chars in View)
     String info = ((TextView) v).getText().toString();
     String address = info.substring(info.length() - 17);
     Toast.makeText(getBaseContext(), "Connection with device "+address, Toast.LENGTH_LONG).show();
   }
 };

// BroadcastReceiver listens for discovered devices and
 // changes the title when discovery is finished
 private final BroadcastReceiver mReceiver = new BroadcastReceiver()  {
   @Override
   public void onReceive(Context context, Intent intent) {
     String action = intent.getAction();
     // When discovery finds a device
     if (BluetoothDevice.ACTION_FOUND.equals(action)) {
       // Get the BluetoothDevice object from the Intent
       BluetoothDevice device =   intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
       // If paired, skip it, because is listed
       if (device.getBondState() != BluetoothDevice.BOND_BONDED) {
         mNewDevicesArrayAdapter.add(device.getName() + "\n" +  device.getAddress());
       }
       // When discovery finished, change Activity title
     }
```



```
      else if
        (BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action)) {
        txt1.setBackgroundResource(R.color.select_background);
        txt1.setText(R.string.select_device);
        if (mNewDevicesArrayAdapter.getCount() == 0) {
         String noDevices = getResources().getText(R.string.none_found).toString();
          mNewDevicesArrayAdapter.add(noDevices);
        }
      }
    }
  };
}
```

MAC address (Media Access Control address) – Δεκαεξαδικός αριθμός 12 ψηφίων ο οποίος είναι μοναδικός για κάθε δικτυακή συσκευή και χρησιμοποιείται για την επικοινωνία μεταξύ των δικτυακών συσκευών ενός τοπικού δικτύου

Σύνδεση Συσκευών



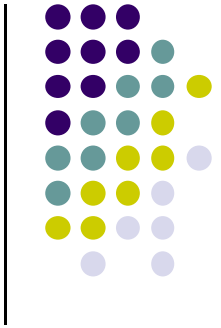
Η επικοινωνία μεταξύ δύο συσκευών γίνεται με τη μέθοδο server-client, κάθε μια από τις οποίες πρέπει να έχει ένα *BluetoothSocket* στο ίδιο κανάλι RFCOMM. Μετά είναι δυνατή η μεταφορά δεδομένων μεταξύ των δύο συσκευών.

Η συσκευή-server δημιουργεί ένα *BluetoothSocket* όταν μια εισερχόμενη αίτηση σύνδεσης γίνει αποδεκτή. Η συσκευή-client δημιουργεί ένα *BluetoothSocket* όταν δημιουργήσει ένα κανάλι RFCOMM προς τον server.

- Η χρησιμοποιούμενη μέθοδος είναι οι δύο συσκευές να χαρακτηριστούν σαν servers ώστε και οι δύο να έχουν ανοικτό ένα server socket και να περιμένουν για αίτηση σύνδεσης. Έτσι, μια από τις δύο συσκευές μπορεί να ξεκινήσει τη διαδικασία σύνδεσης με την άλλη και να γίνει στη συνέχεια client.

- Εναλλακτικά, η μία συσκευή ανοίγει ένα server socket και η άλλη ξεκινάει τη διαδικασία σύνδεσης.

Σύνδεση σαν Server



Η σύνδεση δύο συσκευών απαιτεί η μία από αυτές να λειτουργεί σαν server διαθέτοντας ένα *BluetoothServerSocket*. Σκοπός του server socket είναι να δέχεται αιτήσεις σύνδεσης και όταν κάποια από αυτές γίνει αποδεκτή, να διαθέσει ένα *BluetoothSocket* για σύνδεση. Μετά από αυτό, το *BluetoothServerSocket* αποσυνδέεται εκτός εάν είναι επιθυμητές και άλλες συνδέσεις.

Universally Unique Identifier (UUID)

Το UUID είναι ένα ID μήκους 128 bits για τη μοναδική ταυτοποίηση της πληροφορίας. Χρησιμοποιείται για να προσδιορίζει μοναδικά τη συσκευή BT. Η δημιουργία ενός UUID γίνεται με μια τυχαία γεννήτρια παραγωγής κωδικών.

Σύνδεση σαν Client

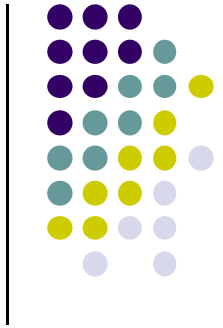
Για τη σύνδεση με μια απομακρυσμένη συσκευή, η οποία διαθέτει ένα ενεργό server socket (συσκευή BT), πρέπει να δημιουργήσουμε ένα αντικείμενο *BluetoothDevice* που αντιπροσωπεύει τη συσκευή (αυτό γίνεται με την αναζήτηση των γειτονικών συσκευών). Με το αντικείμενο αυτό γίνεται προσπάθεια διάθεσης ενός *BluetoothSocket* για να ξεκινήσει η σύνδεση.

Τα βασικά σημεία αυτής της λειτουργίας είναι τα εξής:

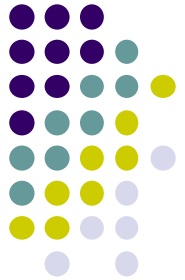
- Χρησιμοποιώντας το αντικείμενο *BluetoothDevice* και με την κλήση της *createRfcommSocketToServiceRecord(UUID)* προσπαθούμε να δημιουργήσουμε ένα *BluetoothSocket*. Το UUID πρέπει να είναι ταυτόσημο με το αντίστοιχο του server, το οποίο δημιουργείται με το *listenUsingRfcommWithServiceRecord(String, UUID)* όταν αυτός δημιουργεί ένα *BluetoothServerSocket* .
- Εκκίνηση της σύνδεσης με την κλήση της ***connect()***. Ο server τότε θα ελέγξει το UUID και εφόσον είναι σωστό θα διαθέσει ένα κανάλι RFCOMM για τη σύνδεση και η διαδικασία θα επιστρέψει από τη μέθοδο *connect()*. Εάν υπάρξει αποτυχία σύνδεσης ή περάσουν περίπου 12 sec το σύστημα θα δημιουργήσει εξαίρεση.

  Για τον παραπάνω λόγο, η διαδικασία πραγματοποιείται σε ξεχωριστό νήμα.

Όταν γίνεται κλήση της *connect()* πρέπει να είμαστε βέβαιοι ότι δεν υπάρχει διαδικασία αναζήτησης συσκευών. Εάν υπάρχει αναζήτηση συσκευών τότε η διαδικασία σύνδεσης είτε θα καθυστερήσει είτε θα αποτύχει.

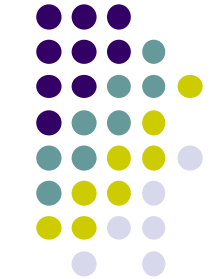


```
private class ConnectThread extends Thread {
  private final BluetoothSocket mmSocket;
  private final BluetoothDevice mmDevice;

  public ConnectThread(BluetoothDevice device) {
    // Use a temporary object that is later assigned to mmSocket,
    // because mmSocket is final
    BluetoothSocket tmp = null;
    mmDevice = device;
    // Get a BluetoothSocket to connect with the given BluetoothDevice
    try {
    // MY_UUID is the app's UUID string, also used by the server code
      tmp = device.createRfcommSocketToServiceRecord(MY_UUID);
    } catch (IOException e) { }
        mmSocket = tmp;
  }

  public void run() {
    // Cancel discovery because it will slow down the connection
    mBluetoothAdapter.cancelDiscovery();
    try {
      // Connect the device through the socket. This will block
      // until it succeeds or throws an exception
      mmSocket.connect();
    } catch (IOException connectException) {
        // Unable to connect; close the socket and get out
        try {
          mmSocket.close();
        } catch (IOException closeException) { }
            return;
    }
```



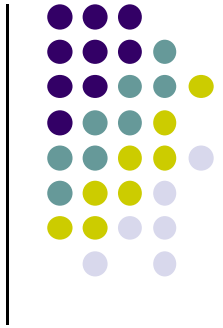
```
    // Do work to manage the connection (in a separate thread)
    connected(mmSocket, mmDevice);
  }


/** Will cancel an in-progress connection, and close the socket */
  public void cancel() {
    try {
      mmSocket.close();
    } catch (IOException e) { }
  }
}
```

Διαχείριση σύνδεσης

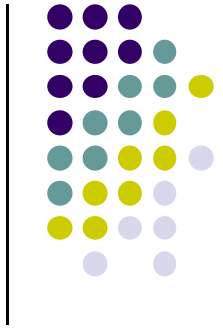


Όταν η σύνδεση μεταξύ δύο συσκευών BT είναι επιτυχής κάθε μια θα διαθέτει ένα *BluetoothSocket*. Αυτό επιτρέπει την ανταλλαγή δεδομένων μεταξύ τους. Η διαδικασία είναι η εξής:

- Η ροή δεδομένων στο socket γίνεται μέσω των αντικειμένων εισόδου και εξόδου *InputStream* και *OutputStream* αντίστοιχα και των μεθόδων *getInputStream() και getOutputStream().*
- Η ανάγνωση ή εγγραφή δεδομένων γίνεται με τις μεθόδους *read(byte[]) και write(byte[]).*
- Η ανάγνωση ή εγγραφή πρέπει να γίνεται σε ξεχωριστό νήμα επειδή οι παραπάνω μέθοδοι μπορεί να μπλοκάρουν το πρόγραμμα (blocking calls). Η μεν ανάγνωση να καθυστερήσει μέχρι να διατεθεί κάτι προς ανάγνωση, η δε εγγραφή να κολλήσει αν οι ενδιάμεσοι buffers είναι γεμάτοι επειδή η άλλη συσκευή δεν διαβάζει γρήγορα. Έτσι ο κύριος βρόχος στο νήμα αφιερώνεται στην ανάγνωση από το *InputStream*, ενώ μια άλλη μέθοδος στο νήμα μπορεί να χρησιμοποιηθεί για να πραγματοποιεί εγγραφές στο *OutputStream*.

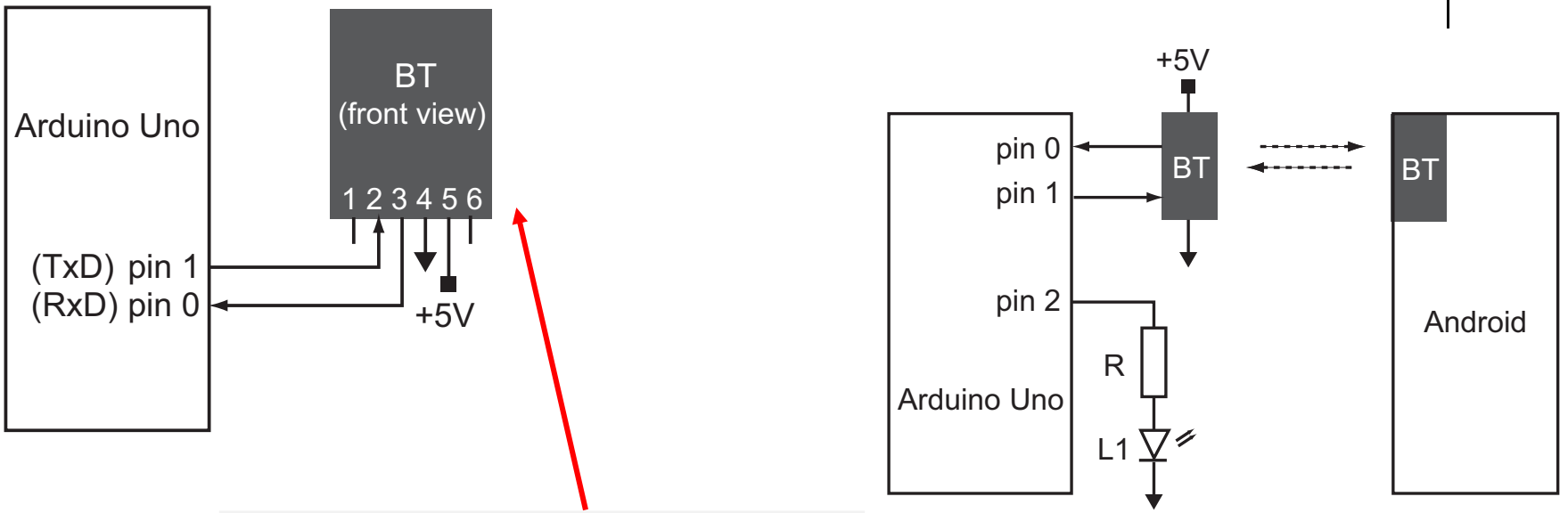


```
private class ConnectedThread extends Thread {
  private final BluetoothSocket mmSocket;
  private final InputStream mmInStream;
  private final OutputStream mmOutStream;

  public ConnectedThread(BluetoothSocket socket) {
    mmSocket = socket;
    InputStream tmpIn = null;
    OutputStream tmpOut = null;
    // Get the input and output streams, using temp objects because
    // member streams are final
    try {
      tmpIn = socket.getInputStream();
      tmpOut = socket.getOutputStream();
    } catch (IOException e) { }
       mmInStream = tmpIn;
       mmOutStream = tmpOut;
  }
  public void run() {
    byte[] buffer = new byte[1024];           // buffer store for the stream
    int bytes; // bytes returned from read()
```



```
 // Keep listening to the InputStream until an exception occurs
  while (true) {
    try {
      // Read from the InputStream
      bytes = mmInStream.read(buffer);
      // Send the obtained bytes to the UI activity
      mHandler.obtainMessage(MESSAGE_READ, bytes, -1, buffer).sendToTarget();
    } catch (IOException e) {
       break;
    }
  }
}

/* Call this from the main activity to send data to the remote device */
public void write(byte[] bytes) {
  try {
    mmOutStream.write(bytes);
    } catch (IOException e) { }
 }

/* Call this from the main activity to shutdown the connection */
public void cancel() {
  try {
    mmSocket.close();
  } catch (IOException e) { }
 }
}
```

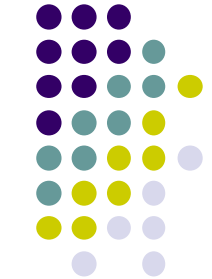
Σύνδεση συσκευών BT



Συσκευή BT

Σειριακή επικοινωνία: 9600 , N, 8, 1
Όνομα: linvor
Κώδικας ζεύγους (Pairing code): 1234
Προγραμματισμένο σαν συσκευή “slave”
Τροφοδοσία 3.3 έως 5 V

Εφαρμογή σύνδεσης

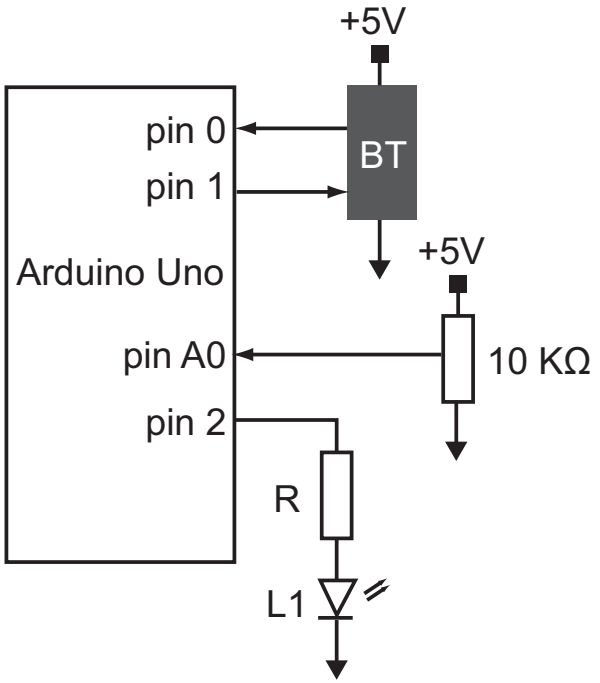


Ανταλλαγή δεδομένων Android με Arduino μέσω Bluetooth. Η εφαρμογή Android θα έχει δύο δραστηριότητες:

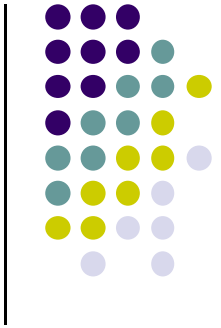
- Στην κύρια δραστηριότητα θα δημιουργείται μια οθόνη χρήστη η οποία θα περιέχει δύο πεδία κειμένου, ένα πλήκτρο εναλλαγής κατάστασης (Toggle Button) και ένα Image Button. Το Image Button περιέχει το σύμβολο επικοινωνίας Bluetooth, το οποίο από κόκκινο μετατρέπεται σε πράσινο όταν επιτευχθεί σύνδεση με μια άλλη συσκευή BT. Το πεδίο κειμένου που βρίσκεται δίπλα στο Image Button περιέχει το όνομα της συσκευής BT με την οποία γίνεται η σύνδεση. Το πλήκτρο εναλλαγής κατάστασης χρησιμοποιείται για να στέλνει στον Arduino τρία bytes για την αλλαγή κατάστασης ενός LED το οποίο είναι συνδεδεμένο στο pin 2. Στο δεύτερο πεδίο κειμένου απεικονίζεται η αναλογική είσοδος A0 του Arduino, στην οποία είναι συνδεδεμένο ένα ποτενσιόμετρο για τη μεταβολή της τάσης εισόδου.
- Η δεύτερη δραστηριότητα καταγράφει τις συσκευές BT που έχουν συνδεθεί κατά το παρελθόν με την κινητή συσκευή καθώς επίσης έχει τη δυνατότητα σάρωσης και καταγραφής νέων συσκευών BT. Μόλις επιλεγεί κάποια απομακρυσμένη συσκευή BT από τον κατάλογο, τότε επιστρέφει στην κύρια δραστηριότητα η οποία αναλαμβάνει να πραγματοποιήσει τη σύνδεση με τη συγκεκριμένη συσκευή.


MyBT_2
Connected BT Device
Arduino Analog Input
OFF


13:30
MyBT_2
linvor
472
OFF

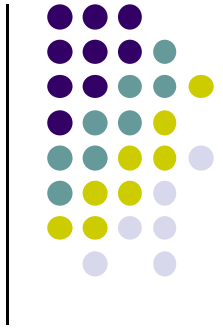

+5V
pin 0
pin 1
BT
Arduino Uno
+5V
pin A0
10 KΩ
pin 2
R
L1

Εφαρμογή Arduino (*bt_2.ino*)



```
// define constants for protocol
#define COMMAND_LED 0x2              //LED driving
#define TARGET_LED 0x2               //LED at pin 2
#define led 2
#define COMMAND_TEMP 0x4    // input from temperature sensor
#define TARGET_TEMP 0x0              // analog pin 0
#define apin 0
#define TOGGLE 0xf

int ledStatus = HIGH;
uint8_t sndmsg[6];

void setup() {
  pinMode(led, OUTPUT);
  Serial.begin(57600);
  Serial.flush();
}

void loop() {
  Serial.flush();
  if (Serial.available()>0){
    int command = Serial.read();
    if (command == COMMAND_LED) {
      int target = Serial.read();
      if (target == TARGET_LED) {
        int toggle = Serial.read();
 if (toggle == TOGGLE) {
        digitalWrite(led, ledStatus);
        ledStatus = (ledStatus == LOW) ? HIGH : LOW;
      }
    }
  }
 }
 int span = 20;
 long analog = 0;
 for (int i = 0; i < span; i++) {
   analog = analog + analogRead(apin);
 }
 analog = analog / 20;
 sndmsg[0] = COMMAND_TEMP;
 sndmsg[1] = TARGET_TEMP;
 sndmsg[2] = (byte) (analog >> 24);
 sndmsg[3] = (byte) (analog >> 16);
 sndmsg[4] = (byte) (analog >> 8);
 sndmsg[5] = (byte) analog;
 Serial.write(sndmsg,6);
}
```

Εφαρμογή Android (MyBT_2)



```
// The on-click listener for all devices in the ListViews
private OnItemClickListener mDeviceClickListener = new  OnItemClickListener() {
  public void onItemClick(AdapterView<?> av, View v, int arg2, long arg3) {
    // Cancel discovery because it is time to connect
    mBluetoothAdapter.cancelDiscovery();

    // Get the device MAC address, which is the last 17 chars in View
    String info = ((TextView) v).getText().toString();
    String address = info.substring(info.length() - 17);
    Intent intent = new Intent();
    intent.putExtra(MainActivity.DEVICE_ADDRESS, address);
    setResult(Activity.RESULT_OK, intent);
    finish();
  }
};
```

Η δεύτερη δραστηριότητα, *DeviceSearchActivity.java*, είναι ίδια με το παράδειγμα αναζήτησης συσκευών BT. Η μόνη διαφορά είναι ότι η επιλογή μιας συσκευής από οποιαδήποτε λίστα όψης τερματίζει τη δραστηριότητα αφού έχει δημιουργήσει μια εκδήλωση πρόθεσης με αποτέλεσμα *RESULT_OK*, ενσωματώνοντας τη διεύθυνση της συσκευής BT (MAC Address). Να σημειωθεί ότι αυτή η δραστηριότητα έχει ενεργοποιηθεί από την κύρια δραστηριότητα, *MainActivity.java*, με εκδήλωση πρόθεσης μέσω της μεθόδου *startActivityForResult()* και παράμετρο την *REQUEST_CONNECT_DEVICE*. Η ανίχνευση της παραμέτρου αυτής και του αποτελέσματος της δεύτερης δραστηριότητας πραγματοποιείται από τη μέθοδο *onActivityResult()*.

```
package gr.mybook.mybt_2;
import ……
public class MainActivity extends Activity {
  // AOA protocol for temperature sensor & LED
  private static final byte COMMAND_TEMP = 0x4;
  private static final byte COMMAND_LED = 0x2;
  private static final byte TARGET_LED = 0x2;
  private static byte TOGGLE = 0xf;

  // Intent request codes
  private static final int REQUEST_CONNECT_DEVICE=1;
  // Messages sent to Handler from Threads
  public static final int MESSAGE_DEVICE_NAME = 2;
  public static final int MESSAGE_FIELD = 3;
  public static final int MESSAGE_READ = 4;
  public static final int MESSAGE_WRITE = 5;

  // Key names to Handler from Threads
  public static final String DEVICE_NAME = "device_name";
  public static final String FIELD = "field";
  public static final String READ = "read";
  public static final String WRITE = "write";
  // Return Intent extra
  public static String DEVICE_ADDRESS = "device_address";
  ImageButton ibtn;;
  ToggleButton tbtn;
  TextView txt1, txt2;

private BluetoothAdapter mBluetoothAdapter=null;
  private static final UUID MY_UUID =
UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
 private ConnectThread mConnectThread;
  private ConnectedThread mConnectedThread;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  ibtn=(ImageButton) findViewById(R.id.imageButton1);
  tbtn=(ToggleButton) findViewById(R.id.toggleButton1);
  txt1=(TextView) findViewById(R.id.textView1);
  txt2=(TextView) findViewById(R.id.textView2);
  mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
}

@Override
public void onResume() {
  super.onResume();
}

@Override
public void onPause() {
  super.onPause();
}

@Override
public void onDestroy() {
  super.onDestroy();
  stop();
}
```

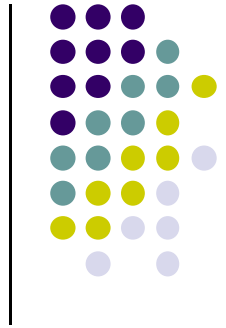
```
public void search(View view) {
  Intent intent = new Intent(this,DeviceSearchActivity.class);
  startActivityForResult(intent, REQUEST_CONNECT_DEVICE);
}

public void onoff(View view) {
  byte[] buffer = new byte[3];
  buffer[0] = COMMAND_LED;
  buffer[1] = TARGET_LED;
  buffer[2] = TOGGLE;
  mConnectedThread.write(buffer);
}

public void onActivityResult(int requestCode, int resultCode, Intent data) {
  switch (requestCode) {
  case REQUEST_CONNECT_DEVICE:
    if (resultCode == Activity.RESULT_OK) {
      // Get the device MAC address
      String address = data.getExtras().getString(DEVICE_ADDRESS);
      // Get the BLuetoothDevice object
      BluetoothDevice device = mBluetoothAdapter.getRemoteDevice(address);
      // Cancel any thread currently running a connection
      if (mConnectedThread != null) {
        mConnectedThread.cancel();
        mConnectedThread = null;
      }
      // Attempt to connect to the device
      mConnectThread = new ConnectThread(device);
      mConnectThread.start();
    }
  }
}
```

Image Button

Toggle Button

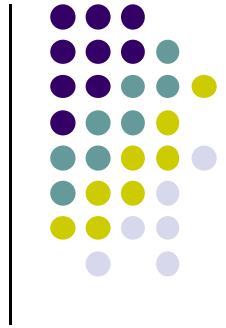
Thread for connection after search & selection of a BT device

```
// Connection attempt failed
 private void connectionFailed() {
   // Send a failure message back to the Activity
   Message msg = mHandler.obtainMessage(MESSAGE_FIELD);
   Bundle bundle = new Bundle();
   bundle.putString(FIELD, "Unable to connect device");
   msg.setData(bundle);
   mHandler.sendMessage(msg);
 }

 // Connection was lost
 private void connectionLost() {
   // Send a failure message back to the Activity
   Message msg = mHandler.obtainMessage(MESSAGE_FIELD);
   Bundle bundle = new Bundle();
   bundle.putString(FIELD, "Device connection was lost");
   msg.setData(bundle);
   mHandler.sendMessage(msg);
 }

 // Thread for connection with remote BT device
 private class ConnectThread extends Thread {
   private final BluetoothSocket mmSocket;
   private final BluetoothDevice mmDevice;
   public ConnectThread(BluetoothDevice device) {
     mmDevice = device;
     BluetoothSocket tmp = null;
     // Get BluetoothSocket for connection with given BluetoothDevice
     try {
       tmp = device.createRfcommSocketToServiceRecord(MY_UUID);
     } catch (IOException e) { }
     mmSocket = tmp;
   }
```



Connection to BT

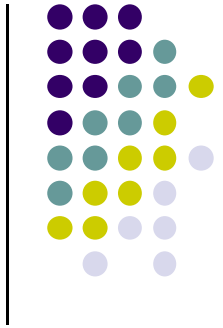
```
public void run() {
  // Always cancel discovery because it slows down a connection
  mBluetoothAdapter.cancelDiscovery();
  // Make a connection to the BluetoothSocket
  try {
    // This is a blocking call and will only return on a
    // successful connection or an exception
    mmSocket.connect();
  } catch (IOException e) {
      connectionFailed();
      // Close the socket
      try {
        mmSocket.close();
      } catch (IOException e2) { }
    return;
  }
  // Reset the ConnectThread-connection ok
  synchronized (MainActivity.this) {
    mConnectThread = null;
  }
  // Start the connected thread
  connected(mmSocket, mmDevice);
 }

 public void cancel() {
  try {
    mmSocket.close();
  } catch (IOException e) { }
 }
}
```



Start another thread for exchange data after connection

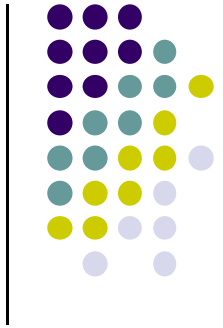
```
// Management of BT connection through ConnectedThread
 public void connected(BluetoothSocket socket, BluetoothDevice device){
   // Cancel the thread that completed the connection
   if (mConnectThread != null) {
     mConnectThread.cancel();
     mConnectThread = null;
   }
   // Cancel any thread currently running a connection
   if (mConnectedThread != null) {
     mConnectedThread.cancel();
     mConnectedThread = null;
   }
   // Start thread to manage connection and perform transmissions
   mConnectedThread = new ConnectedThread(socket);
   mConnectedThread.start();
   // Send the name of the connected device back to the UI Activity
   Message msg = mHandler.obtainMessage (MESSAGE_DEVICE_NAME);
   Bundle bundle = new Bundle();
   bundle.putString(DEVICE_NAME, device.getName());
   msg.setData(bundle);
   mHandler.sendMessage(msg);
 }
 // Stop all threads
 public void stop() {
   if (mConnectThread != null) {
     mConnectThread.cancel();
     mConnectThread = null;
   }
   if (mConnectedThread != null) {
     mConnectedThread.cancel();
     mConnectedThread = null;
   }
 }
```

Start another thread for exchange data after connection



```
// Manage connection with BT device - in and out data
private class ConnectedThread extends Thread {
  private final BluetoothSocket mmSocket;
  private final InputStream mmInStream;
  private final OutputStream mmOutStream;
  public ConnectedThread(BluetoothSocket socket) {
    mmSocket = socket;
    InputStream tmpIn = null;
    OutputStream tmpOut = null;
    // Get the BluetoothSocket input and output streams
    try {
      tmpIn = socket.getInputStream();
      tmpOut = socket.getOutputStream();
    } catch (IOException e) { }
    mmInStream = tmpIn;
    mmOutStream = tmpOut;
  }
  public void run() {
    byte[] buffer = new byte[1024];
    int bytes;
    // Keep listening to the InputStream while connected
    while (true) {
      try {
        // Read from the InputStream
        bytes = mmInStream.read(buffer);
        // Send the obtained bytes to the UI Activity
        mHandler.obtainMessage(MESSAGE_READ, bytes, -1, buffer).sendToTarget();
      } catch (IOException e) {
          connectionLost();
          break;
        }
    }
  }
```

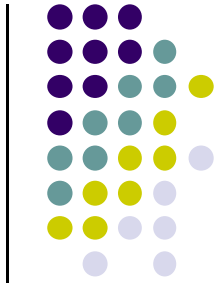
Thread for exchange data through Handler



```
// Write to the connected OutStream
   public void write(byte[] buffer) {
     try {
       mmOutStream.write(buffer);
       // Share the sent message back to the UI Activity
       mHandler.obtainMessage(MESSAGE_WRITE, buffer.length, -1, buffer).sendToTarget();
     } catch (IOException e) { }
   }

   public void cancel() {
     try {
       mmSocket.close();
     } catch (IOException e) { }
   }
 }
// The Handler gets information back
 private final Handler mHandler = new Handler() {
   @Override
   public void handleMessage(Message msg) {
     switch (msg.what) {
     case MESSAGE_WRITE:
       break;
     case MESSAGE_READ:
       byte[] buffer = (byte[]) msg.obj;
       int length=msg.arg1;
       if (buffer[0] == COMMAND_TEMP) {
         final int temp = (((buffer[2] & 0xff) << 24)
                         + ((buffer[3] & 0xff) << 16)
                         + ((buffer[4] & 0xff) << 8)
                         + (buffer[5] & 0xff));
         String mytext=Integer.toString(temp);
         txt2.setText(mytext);
       }
       break;
     case MESSAGE_DEVICE_NAME:
       // save the connected device's name
       txt1.setText(msg.getData().getString(DEVICE_NAME));
       int bgrnd = getResources().getColor(R.color.green);
       ibtn.setBackgroundColor(bgrnd);
       break;
     case MESSAGE_FIELD:
       Toast.makeText(MainActivity.this, msg.getData().
                 getString(FIELD),Toast.LENGTH_SHORT).show();
       break;
     }
   }
  };
}
```

Handler to READ & WRITE data

Ασκήσεις



- Δημιουργείστε δύο πλήκτρα ON-OFF στην οθόνη του κινητού και αναβοσβήνετε τα δύο LED που υπάρχουν στο shield αισθητηρίων μέσω Bluetooth ακολουθώντας το παράδειγμα της ενότητας αυτής.