

ANDROID İLE ARDUINO

NAZİR DOĞAN

İÇİNDEKİLER

BÖLÜM 1: BAŞLARKEN	1
ADK Nedir?	2
Donanım Geliştirme Kartları	4
Google ADK 2011	4
Arduino ADK	6
Google ADK 2012	6
Arduino DUE	7
IOIO	8
Desteklenen Android Cihazlar	9
BÖLÜM 2: GELİŞTİRME ORTAMININ KURULUMU	11
Java Geliştirme Kiti (JDK)	12
Windows için Kurulum	13
Linux için Kurulum	16
Mac OSX için Kurulum	17
Android SDK	17
Arduino IDE	19
Windows Kurulumu	21
Windows 8 Kurulumu	24
Linux Kurulumu	26
Mac OSX Kurulumu	27
Fritzing Nedir?	28
Processing	28
BÖLÜM 3: ANDROID VE ARDUINO'YU TANIŞTIRMA	31
Merhaba Arduino Uygulaması	32
Merhaba Android Uygulaması	35
Uygulama Bileşenleri	36

ADK için Arduino Merhaba Dünya Uygulaması	38
ADK için Arduino Merhaba Dünya Uygulaması	49
Arduino'ya Kütüphane Ekleme	49
Windows ve Mac	49
Linux	50
Arduino Kodları	54
Arduino ADK ile Android'i Konuşturma	54
BÖLÜM 4: ÇIKIŞLAR (OUTPUTS)	71
Proje 1 - Led Yakma	72
Devre Kurulumu	72
Yazılım	74
Android Uygulaması	75
Proje 2 - Led'in Işık Şiddetini Ayarlama	82
Devre Kurulumu	83
Yazılım	83
Android Uygulaması	85
BÖLÜM 5: GİRİŞLER (INPUTS)	93
Buton Durumu Okuma	94
Kurulum	94
Android Uygulaması	96
Kurulum	103
Yazılım	104
Android	106
BÖLÜM 6: IŞIK ŞİDDETİ ÖLÇME	113
Proje 5 - Foto Direnç (LDR) ile Işık Şiddeti Ölçme	114
Devre Kurulumu	116

Yazılım	116
Arduino	116
Android Uygulaması	118
BÖLÜM 7: ARDUINO VE ARDUINO İLE BLUETOOTH İLETİŞİMİ	125
Processing ile Android	126
Processing'e Android Modu Ekleme	126
Ketai Kütüphanesi	127
Proje 5 Bluetooth ile Led Yakma ve Seri Haberleşme	128
HC-06 Bluetooth Modülü	128
Kurulum	129
Yazılım	129
KİTAPTA KULLANILAN MALZEMELER	135
İndeks	136

ÇIKIŞLAR (OUTPUTS)

BU BÖLÜMDE

Proje 1 - Led Yakma	72
Proje 2 - Led'in Işık Şiddetini Ayarlama	82

Orjinal Arduino tasarımından türetilmiş bir ADK kartının bir çok pine ve konnektöre sahip olduğunu önceki bölümlerde söylemiştik. ADK kartının büyük kısmı dijital pinlerden oluşmaktadır. Bu dijital pinler dijital giriş veya çıkış olarak kullanılma yeteneğine sahiptir.

Bu bölümde dijital çıkış pinlerini nasıl kullanacağımızı öğreneceğiz.

Bu Bölümde Kullanılan Malzemeler

- » Arduino Mega ADK (Android Geliştirme Kartı)
- » Asus Memo Pad 7
- » Devre Tahtası
- » USB B tipi kablo (Yazıcı kablosu olarakta bilinir)
- » Micro USB kablo
- » Devre Tahtası (BreadBoard)
- » Jumper Kablolar
- » LED

ANDROİD UYGULAMASI

Android için daha önce yaptığımız gibi *AndroidManifest.xml*'i düzenlemek gerekir. Hatırlayacağınız üzere Android cihazımızın USB özelliğini kullanabilmek için bu düzenlemeye şiddetle ihtiyacımız vardır. Aksi takdirde hiçbir uygulamamız çalışmayacaktır. Ayrıca veri protokolünü ayarlayıp ve LED'i açıp kapatacak bir Arayüz elemanı olan `ToggleButton`'u eklemek lazımdır. Şimdi koda bakalım:

NOT

Android uygulamaları için kullandığımız kaynak kodları uzun olduğundan, bu kodları kitabın başında verdiğim **GitHub** adresinden indirip kullanabilirsiniz.

```
package com.example.projeled;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import java.io.FileDescriptor;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.AsyncTask;
import android.os.ParcelFileDescriptor;
import android.util.Log;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.ToggleButton;
import com.android.future.usb.UsbAccessory;
import com.android.future.usb.UsbManager;

public class MainActivity extends ActionBarActivity {

    private static final String TAG = MainActivity.class.getSimpleName();
    private PendingIntent mPermissionIntent;
    private static final String ACTION_USB_PERMISSION = "com.android.example.USB_PERMISSION";
    private boolean mPermissionRequestPending;
    private UsbManager mUsbManager;
    private UsbAccessory mAccessory;
    private ParcelFileDescriptor mFileDescriptor;
    private FileInputStream mInputStream;
    private FileOutputStream mOutputStream;
```

```
mAccessory = accessory;
FileDescriptor fd = mFileDescriptor.getFileDescriptor();
mInputStream = new FileInputStream(fd);
mOutputStream = new FileOutputStream(fd);
Log.d(TAG, "Aksesuar açıldı");
} else {
    Log.d(TAG, "Aksesuar açılmadı");
}
}
}
private void closeAccessory() {
    try {
        if (mFileDescriptor != null) {
            mFileDescriptor.close();
        }
    } catch (IOException e) {} finally {
        mFileDescriptor = null;
        mAccessory = null;
    }
}
public void sendLedSwitchCommand(byte target, boolean isSwitchedOn) {
    byte[] buffer = new byte[3];
    buffer[0] = COMMAND_LED;
    buffer[1] = target;
    if (isSwitchedOn) {
        buffer[2] = VALUE_ON;
    } else {
        buffer[2] = VALUE_OFF;
    }
    if (mOutputStream != null) {
        try {
            mOutputStream.write(buffer);
        } catch (IOException e) {
            Log.e(TAG, "Yazma başarısız oldu", e);
        }
    }
}
}
```

LED'i kontrol etmek için Arduino Sketch'inde bulunan sabitleri Android uygulama tarafında tanımlamamız gerekmektedir.

```
private static final byte COMMAND_LED = 0x2;
private static final byte TARGET_PIN_2 = 0x2;
private static final byte VALUE_ON = 0x1;
private static final byte VALUE_OFF = 0x0;
```

Bu projemizde herhangi bir metin görüntülemeye ihtiyacımız olmadığı için Projenin oluşturduğu textView'i silip LED'i açıp kapatmak için bir tane ToggleButton'u ekledik.

```
private ToggleButton ledToggleButton;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    mUsbManager = UsbManager.getInstance(this);
    mPermissionIntent = PendingIntent.getBroadcast(this, 0, new Intent(ACTION_USB_PERMISSION), 0);
    IntentFilter filter = new IntentFilter(ACTION_USB_PERMISSION);
    filter.addAction(UsbManager.ACTION_USB_ACCESSORY_DETACHED);
    registerReceiver(mUsbReceiver, filter);
    setContentView(R.layout.activity_main);
    ledToggleButton = (ToggleButton) findViewById(R.id.led_toggle_button);
    ledToggleButton.setOnCheckedChangeListener(toggleButtonCheckedChangeListener);
}
```

ledToggleButton'una eklediğimiz OnCheckedChangeListener olayı butona her basıldığında tetiklenir. ToggleButton, Button elementinin özelleşmiş halidir. Seçili mi değil mi, kontrolünü yapabilirsiniz.

OnCheckedChangeListener olayı içerisinde bulunan onCheckedChangeListener adında iki tane parametre alan bir metod bulunmaktadır. Butonun durumu kontrol edilerek yeni durum ona göre ayarlanır.

```
OnCheckedChangeListener toggleButtonCheckedChangeListener = new OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (buttonView.getId() == R.id.led_toggle_button) {
            new AsyncTask<Boolean, Void, Void>() {
                @Override
                protected Void doInBackground(Boolean... params) {
```

```

        sendLedSwitchCommand(TARGET_PIN_2, params[0]);
        return null;
    }
    }.execute(isChecked);
}
}
};

```

sendLedSwitchCommand metodu daha önce yaptığımız örnekte kullandığımız sendText metodu ile hemen hemen aynıdır. Burada ise 3 bayt data göndermektedir:

```

public void sendLedSwitchCommand(byte target, boolean isSwitchedOn) {
    byte[] buffer = new byte[3];
    buffer[0] = COMMAND_LED;
    buffer[1] = target;
    if (isSwitchedOn) {
        buffer[2] = VALUE_ON;
    } else {
        buffer[2] = VALUE_OFF;
    }
    if (mOutputStream != null) {
        try {
            mOutputStream.write(buffer);
        }
        catch (IOException e) {
            Log.e(TAG, "Yazma başarısız oldu", e);
        }
    }
}

```

Bütün bu kodları ekledikten sonra ise arayüz için değişiklik yapmamız gerekmektedir. *activity_main.xml* açıp **Grafik Designer**'a geçtikten sonra bunu görmemiz lazım:

NOT

ADT (*Android Developer Tools*) uygulamalarımızın arayüzünü düzenlememiz için **Grafik Designer**'a sahiptir. **Grafik Designer**'da bırak ile yeni arayüz elemanları ekleyebiliriz. Bunların yerini görsel olarak görebiliriz. Bütün stil özelliklerine mühadele edip değiştirebiliriz.

Arduino Sketch:

```

#include <Max3421e.h>
#include <Usb.h>
#include <AndroidAccessory.h>

#define COMMAND_LED 0x2
#define TARGET_PIN_2 0x2
#define PIN 2

AndroidAccessory acc("Uretici", "Model", "Aciklama", "Versiyon", "URI", "Seri");
byte rcvmsg[3];
void setup() {
    Serial.begin(19200);
    acc.powerOn();
    pinMode(PIN, OUTPUT);
}
void loop() {
    if (acc.isConnected()) { // alınan veriyi byte dizisinde okumak için
        int len = acc.read(rcvmsg, sizeof(rcvmsg), 1);
        if (len > 0) {
            if (rcvmsg[0] == COMMAND_LED) {
                if (rcvmsg[1] == TARGET_PIN_2) { // analog değeri alma
                    byte value = rcvmsg[2]; // çıkış pinini analog değere göre ayarlamak için
                    analogWrite(PIN, value);
                }
            }
        }
    }
}

```

LED'in durumunu belirten (VALUE_ON/VALUE_OFF) sabitlerini sildik. Çünkü artık dijital değerlerle değil analog değerlerle uğraşıyoruz.

Android uygulamasından gönderdiğimiz byte değerleri Arduino Sketch'inde bulunan analogWrite metoduna yazılır. analogWrite metodu dijital pinleri tetikleyerek kare dalgalar üretir. Bizim pinden gönderdiğimiz 0 ve 5V arası değere göre 0 ile 255 arasında değerler alır.

NOT

Dijital değerler ile çalışırken iki tane durumumuz vardır her zaman. Bu durumlar 1 (HIGH) ve 0 (LOW) lardan oluşmaktadır. Ama analog değerler ile çalışırken bu durumlar bizim ihtiyacımızı karşılamamaktadır. Çünkü analog değerler ara değerleri de içermektedir. Analog dünyamızı dijital ortamda anlamlandırmak için PWM tekniğinden faydalanmaktayız.

ANDROİD UYGULAMASI

Android uygulaması **Proje 1**'deki uygulama ile hemen hemen aynı olmakla birlikte yeni olarak SeekBar elementi eklenmiştir. **Proje 1**'i biraz değiştirerek uygulamamızı yapacağız. Uygulamamızın kodu GitHub'da var.

Şimdi koda bir göz atalım:

```
package project.two.adk;
import java.io.FileDescriptor;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import android.app.Activity;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.ParcelFileDescriptor;
import android.util.Log;
import android.widget.SeekBar;
import android.widget.SeekBar.OnSeekBarChangeListener;
import android.widget.TextView;
import com.android.future.usb.UsbAccessory;
import com.android.future.usb.UsbManager;

public class ProjectTwoActivity extends Activity {
    private static final String TAG = ProjectTwoActivity.class.getSimpleName();
    private PendingIntent mPermissionIntent;
    private static final String ACTION_USB_PERMISSION = "com.android.example.USB_PERMISSION";
    private boolean mPermissionRequestPending;
    private UsbManager mUsbManager;
    private UsbAccessory mAccessory;
    private ParcelFileDescriptor mFileDescriptor;
    private FileInputStream mInputStream;
    private FileOutputStream mOutputStream;
    private static final byte COMMAND_LED = 0x2;
    private static final byte TARGET_PIN_2 = 0x2;
    private TextView ledIntensityTextView;
    private SeekBar ledIntensitySeekBar;
    /** Activity ilk oluşturulduğunda çağrılır */@Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```

        mUsbManager = UsbManager.getInstance(this);
        mPermissionIntent = PendingIntent.getBroadcast(this, 0, new Intent(ACTION_USB_PERMISSION), 0);
        IntentFilter filter = new IntentFilter(ACTION_USB_PERMISSION);
        filter.addAction(UsbManager.ACTION_USB_ACCESSORY_DETACHED);
        registerReceiver(mUsbReceiver, filter);
        setContentView(R.layout.main);
        ledIntensityTextView = (TextView) findViewById(R.id.led_intensity_text_view);
        ledIntensitySeekBar = (SeekBar) findViewById(R.id.led_intensity_seek_bar);
        ledIntensitySeekBar.setOnSeekBarChangeListener(ledIntensityChangeListener);
        ledIntensityTextView.setText("LED intensity: " + ledIntensitySeekBar.getProgress());
    }
    /**
     * onCreate() methodundan sonra Activity bekleme durumundan devam
     * ettirildiğinde çağrılır.
     */
    @Override
    public void onResume() {
        super.onResume();
        if (mInputStream != null && mOutputStream != null) {
            return;
        }
        UsbAccessory[] accessories = mUsbManager.getAccessoryList();
        UsbAccessory accessory = (accessories == null ? null : accessories[0]);
        if (accessory != null) {
            if (mUsbManager.hasPermission(accessory)) {
                openAccessory(accessory);
            } else {
                synchronized(mUsbReceiver) {
                    if (!mPermissionRequestPending) mUsbManager.requestPermission(accessory,
                        mPermissionIntent);
                    mPermissionRequestPending = true;
                }
            }
        }

    }

    } else {
        Log.d(TAG, "mAccessory is null");
    }
}
/** Activity sistem tarafından durdurulduğunda çağrılır. */
@Override
public void onPause() {
    super.onPause();
    closeAccessory();
    // Activity'e ihtiyaç duyulmadığında sistem tarafından yok edilirken çağrılır.
}
@Override
public void onDestroy() {
    super.onDestroy();
}

```

```

        unregisterReceiver(mUsbReceiver);
    }
    OnSeekBarChangeListener ledIntensityChangeListener = new OnSeekBarChangeListener() { @Override
    public void onProgressChanged(SeekBar seekBar, int progress,
    boolean fromUser) {
        ledIntensityTextView.setText("LED intensity: " + ledIntensitySeekBar.getProgress());
        new AsyncTask < Byte, Void, Void > () { @Override
        protected Void doInBackground(Byte...params) {
            sendLedIntensityCommand(TARGET_PIN_2, params[0]);
            return null;
        }
        }.execute((byte) progress);
    }
}

@Override
public void onStartTrackingTouch(SeekBar seekBar) {

    // not implemented
}
@Override
public void onStopTrackingTouch(SeekBar seekBar) {

    // not implemented
}
};

private final BroadcastReceiver mUsbReceiver = new BroadcastReceiver() { @Override
public void onReceive(Context context, Intent intent) {
    String action = intent.getAction();
    if (ACTION_USB_PERMISSION.equals(action)) {
        synchronized(this) {
            UsbAccessory accessory = UsbManager.getAccessory(intent);
            if (intent.getBooleanExtra(

                UsbManager.EXTRA_PERMISSION_GRANTED, false)) {
                openAccessory(accessory);
            } else {
                Log.d(TAG, "permission denied for accessory " + accessory);
            }
        }

        mPermissionRequestPending = false;
    }

} else if (UsbManager.ACTION_USB_ACCESSORY_DETACHED.equals(action)) {
    UsbAccessory accessory = UsbManager.getAccessory(intent);
    if (accessory != null && accessory.equals(mAccessory)) {
        closeAccessory();
    }
}
}

```

```
    }  
  }  
};  
private void openAccessory(UsbAccessory accessory) {  
    mFileDescriptor = mUsbManager.openAccessory(accessory);  
    if (mFileDescriptor != null) {  
        mAccessory = accessory;  
        FileDescriptor fd = mFileDescriptor.getFileDescriptor();  
        mInputStream = new FileInputStream(fd);  
        mOutputStream = new FileOutputStream(fd);  
        Log.d(TAG, "accessory opened");  
  
    } else {  
        Log.d(TAG, "accessory open fail");  
  
    }  
}  
private void closeAccessory() {  
    try {  
        if (mFileDescriptor != null) {  
            mFileDescriptor.close();  
        }  
    } catch (IOException e) {  
  
    } finally {  
  
        mFileDescriptor = null;  
        mAccessory = null;  
    }  
}  
public void sendLedIntensityCommand(byte target, byte value) {  
    byte[] buffer = new byte[3];  
    buffer[0] = COMMAND_LED;  
    buffer[1] = target;  
    buffer[2] = value;  
    if (mOutputStream != null) {  
  
        try {  
            mOutputStream.write(buffer);  
        } catch (IOException e) {  
            Log.e(TAG, "yazma hatasi", e);  
  
        }  
    }  
}
```

Kodu verdikten sonra ise şimdi **Proje 1'e** göre değişen yerlere bir göz atalım.

LED'in durumunu belirttiğimiz **ToggleButton**'unu kaldırdık ve onun yerine parlaklığı ayarlamamız için **SeekBar** elementi ve seçtiğimiz parlaklığı göstermek için bir **TextView** elementi ekledik.

```
private TextView ledIntensityTextView;
private SeekBar ledIntensitySeekBar; /** Activity ilk oluşturulduğunda çağrılır. */
@Override public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    mUsbManager = UsbManager.getInstance(this);
    mPermissionIntent = PendingIntent.getBroadcast(this, 0, new Intent(ACTION_USB_PERMISSION), 0);
    IntentFilter filter = new IntentFilter(ACTION_USB_PERMISSION);
    filter.addAction(UsbManager.ACTION_USB_ACCESSORY_DETACHED);
    registerReceiver(mUsbReceiver, filter);
    setContentView(R.layout.main);
    ledIntensityTextView = (TextView) findViewById(R.id.led_intensity_text_view);
    ledIntensitySeekBar = (SeekBar) findViewById(R.id.led_intensity_seek_bar);
    ledIntensitySeekBar.setOnSeekBarChangeListener(ledIntensityChangeListener);
    ledIntensityTextView.setText("LED parlaklığı: " + ledIntensitySeekBar.getProgress());
}
```

SeekBar elementine **OnSeekBarChangeListener** olayı ekleyerek değeri değiştğinde **Arduino** tarafına gönderir.

```
OnSeekBarChangeListener ledIntensityChangeListener = new OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
        ledIntensityTextView.setText("LED intensity: " + ledIntensitySeekBar.getProgress());
        new AsyncTask < Byte, Void, Void > () { @Override protected Void doInBackground(Byte...params) {
            sendLedIntensityCommand(TARGET_PIN_2, params[0]);
            return null;
        }
        }.execute((byte) progress);
    } @Override public void onStartTrackingTouch(SeekBar seekBar) { // not implemented
    } @Override public void onStopTrackingTouch(SeekBar seekBar) { // not implemented
    }
};
```

ARDUINO VE ARDUINO İLE BLUETOOTH İLETİŞİMİ

BU BÖLÜMDE

Processing ile Android	126
Proje 5 Bluetooth ile Led Yakma ve Seri Haberleşme	128

Bu bölümde Arduino ve Android'in BlueTooth kullanarak birbiri ile haberleşmesini sağlayacağız. Örneklerimizi **Processing** ortamı ile geliştireceğiz.

Bu bölümde uygulayacağımız örnekleri gayet basit ve anlaşılabilir tuttum. Bu sayede, kendi başınıza yapacağınız uygulamalar için iyi bir referans oluşturmasını arzu ediyorum.

Bu Bölümde Kullanılan Malzemeler

- » Arduino Mega ADK (Android Geliştirme Kartı)
- » Asus Memo Pad 7
- » Devre Tahtası
- » USB B tipi kablo (Yazıcı kablosu olarakta bilinir)
- » Micro USB kablo
- » Devre Tahtası (BreadBoard)
- » Jumper Kablolar
- » HC-06 Bluetooth Modülü

Bluetooth 1994 yılında Ericsson firması tarafından cep telefonları ve diğer cihazları kablosuz olarak birbirine bağlamak amacıyla geliştirilmiştir. Bugün akıllı telefonlarımızın birçoğunda bulunmaktadır. Ayrıca Bluetooth üzerinden haberleşen aksesuarlar da bulunmaktadır. Örneğin Bluetooth Hoparlörler, akıllı saatler vb..

Bu bölümde Arduino ve Android'in Bluetooth kullanarak birbiri ile haberleşmesini sağlayacağız. Bluetooth teknolojisini kullanmamızın nedeni kablosuz kısa mesafeli, ses ve veri haberleşmesini sağlamak için oluşturulmuş bir standard olmasıdır. Bluetooth standardı 10 metreye kadar ses ve veri aktarımını sağlamayı öngörmektedir. Mobil cihazlarda çok önemli bir husus olan güç tüketimi açısından kıyasladığımızda ise Bluetooth, Wi-Fi'ye göre daha az enerji tüketmektedir.

Örneğimizi diğer projelerin aksine Processing ortamı ile geliştireceğiz. Oldukça kolay bir örnek ile konuyu anlamaya çalışacağız.

PROCESSING İLE ANDROID



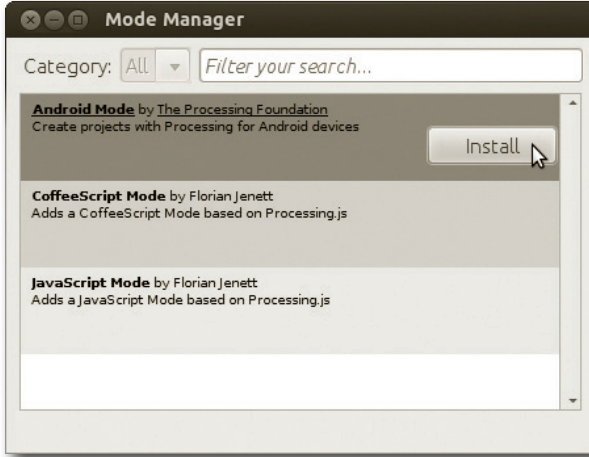
Android SDK ve **Processing** kurulumunu **Bölüm 2**'de ele almıştık. Bu bölümde Android modunu ekleyerek projelerimize başlayacağız.

PROCESSING'E ANDROID MODU EKLEME

Processing'de Android uygulamaları yazmak için Android modunu eklememiz gerekmektedir. Bunun için Processing ortamını açıp sağ üst köşede **Java** yazan kısma tıklıyoruz. Tıkladığımızda küçük bir menü açılacak. Açılan menüde **Add Mode**'a tıklıyoruz.



Tıkladıktan sonra aşağıdaki menü açılacaktır.



Açılan menüde **Android Mode**'e **Install** butonuna tıklayarak kuruyoruz.

Android Mode'u indirdikten sonra **Processing** ortamında gözükmeyecektir. Bunun için Processing'i yeniden başlatmamız gerekmektedir. Yeniden başlattıktan sonra **Android Mode**'u sağ üst köşedeki menüye gelecektir. Menüde Android'e tıkladıktan sonra kurulumu ilk defa yaptığımız için, Processing bizden Android SDK'nin yerini göstermemizi isteyecektir.



Locate SDK path manually diyerek Android SDK'nin yolunu ayarlıyoruz.

KETAI KÜTÜPHANESİ

Bu kütüphane Android telefon ve tabletlerin sensörlerini ve yerleşik donanımlarıyla çalışmayı kolaylaştırmak amacıyla yazılmıştır. Özellikle Bluetooth, Wifi ve NFC iletişimi, çoklu dokunmatik ve sensörle çalışmayı oldukça kolay bir hale