

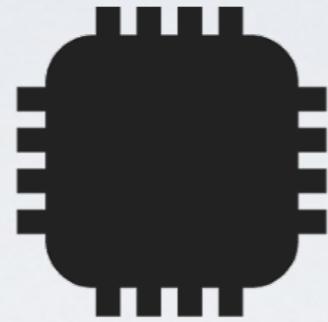


Kart Programming

François Corthay | Charles Praplan

Michael Clausen | Adam True

Goals

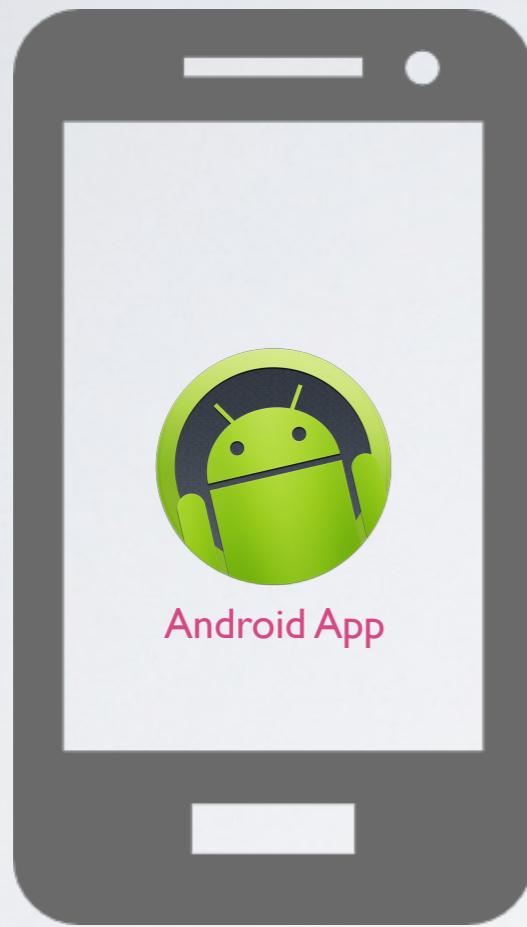


Use basic elements seen in ELN course.



Use basic elements seen in INF I course.

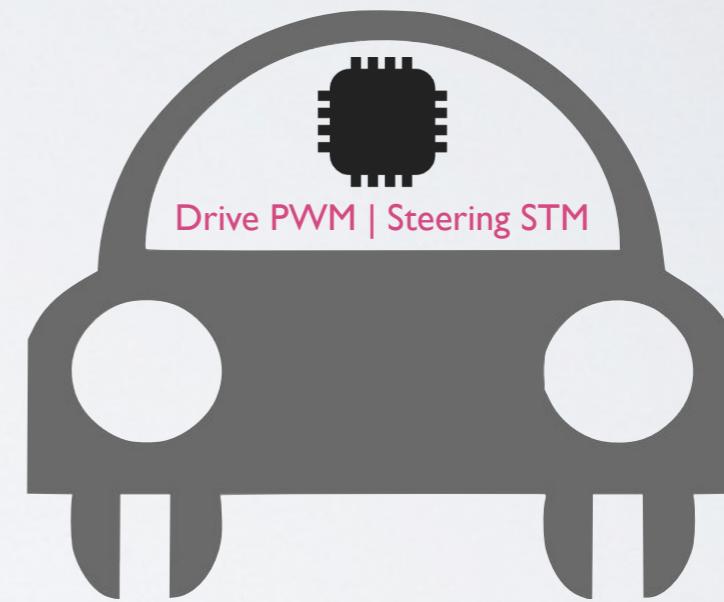
Goals



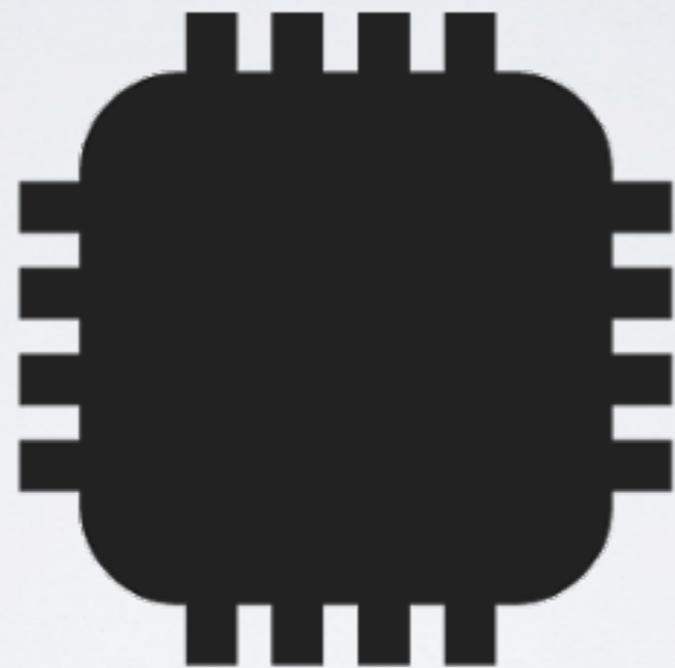
Galaxy Nexus



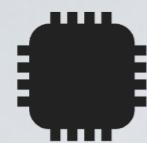
Bluetooth



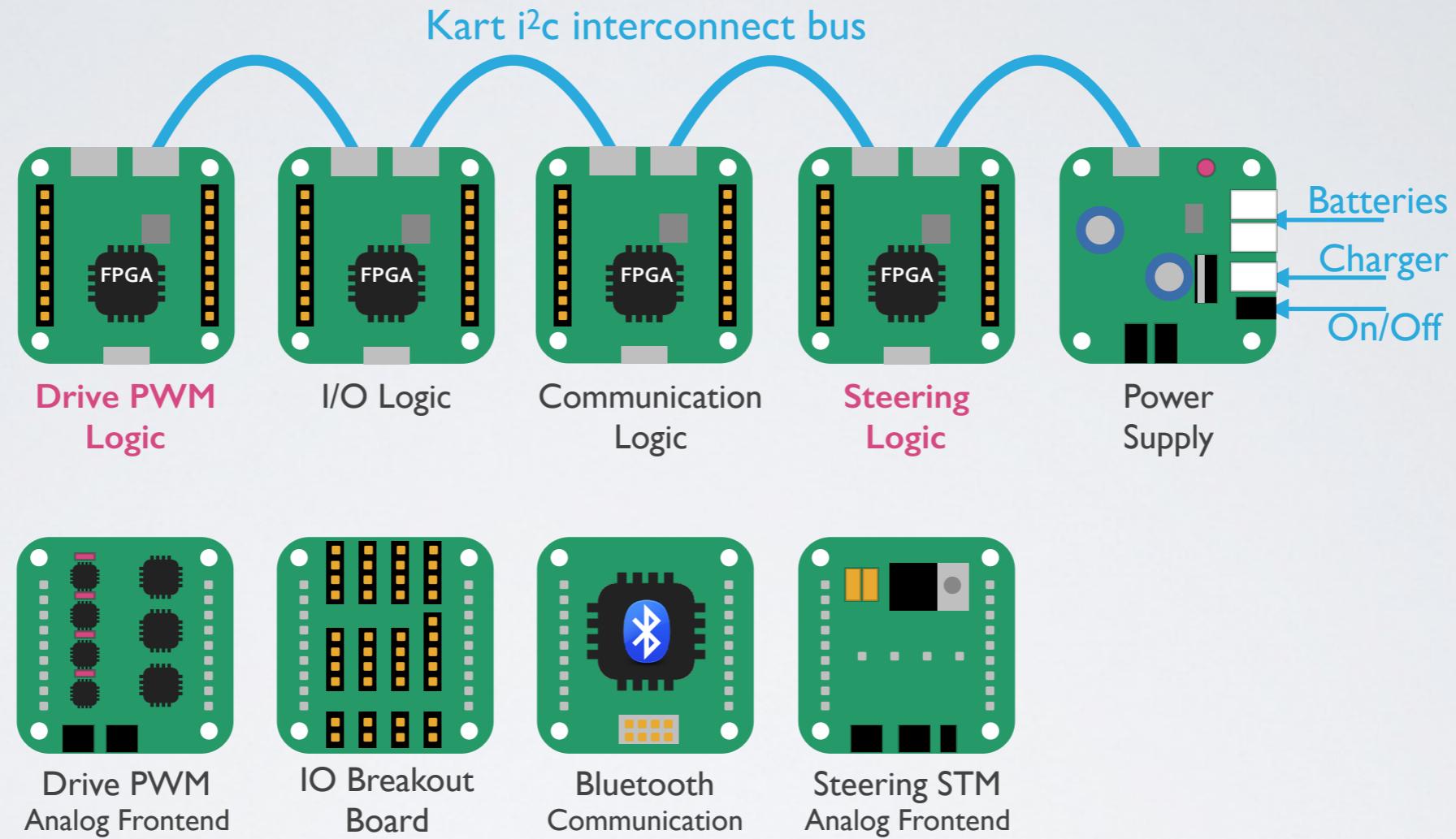
Kart

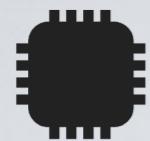


ELN part

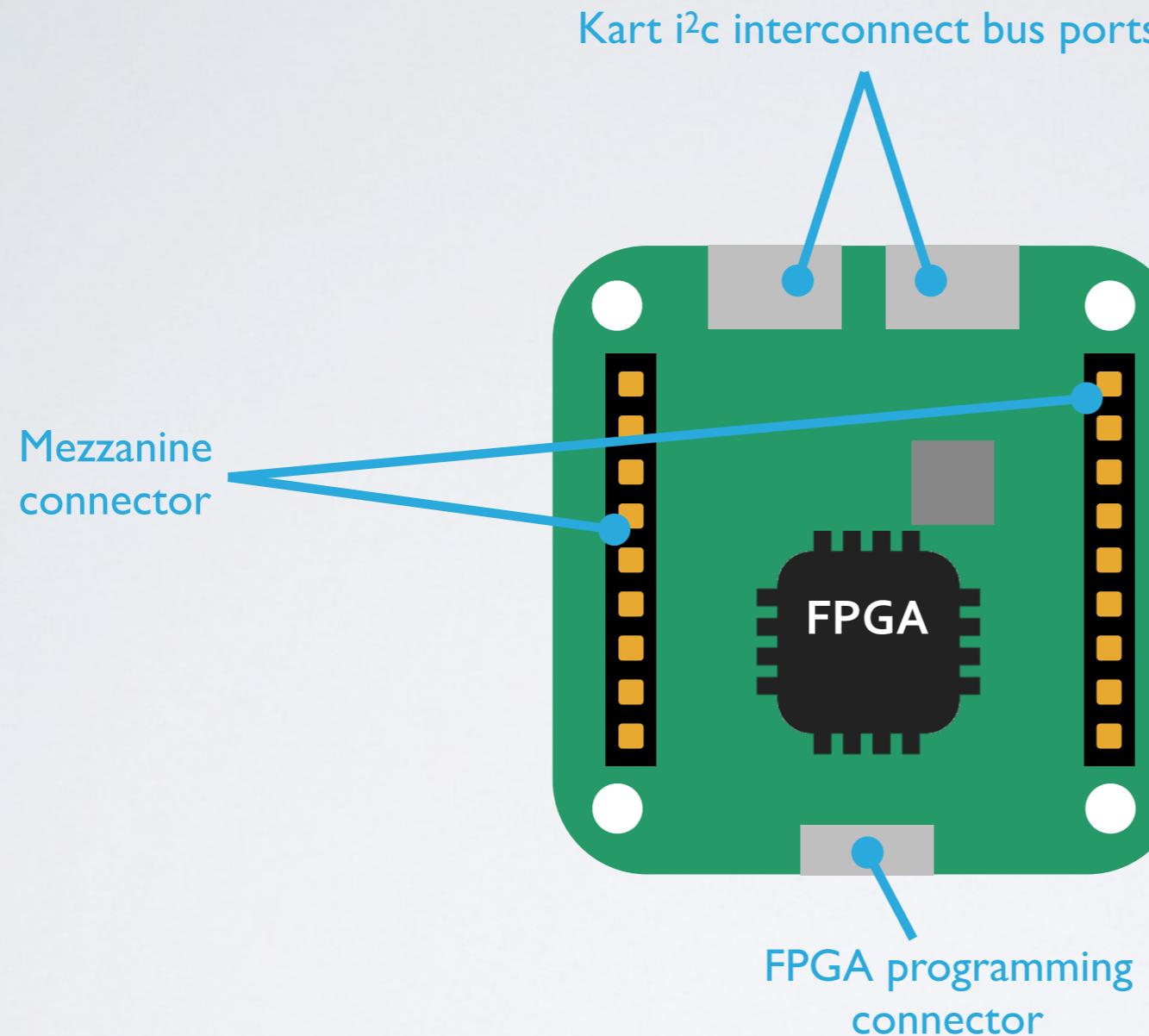


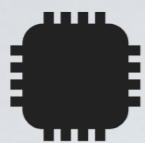
Modular concept





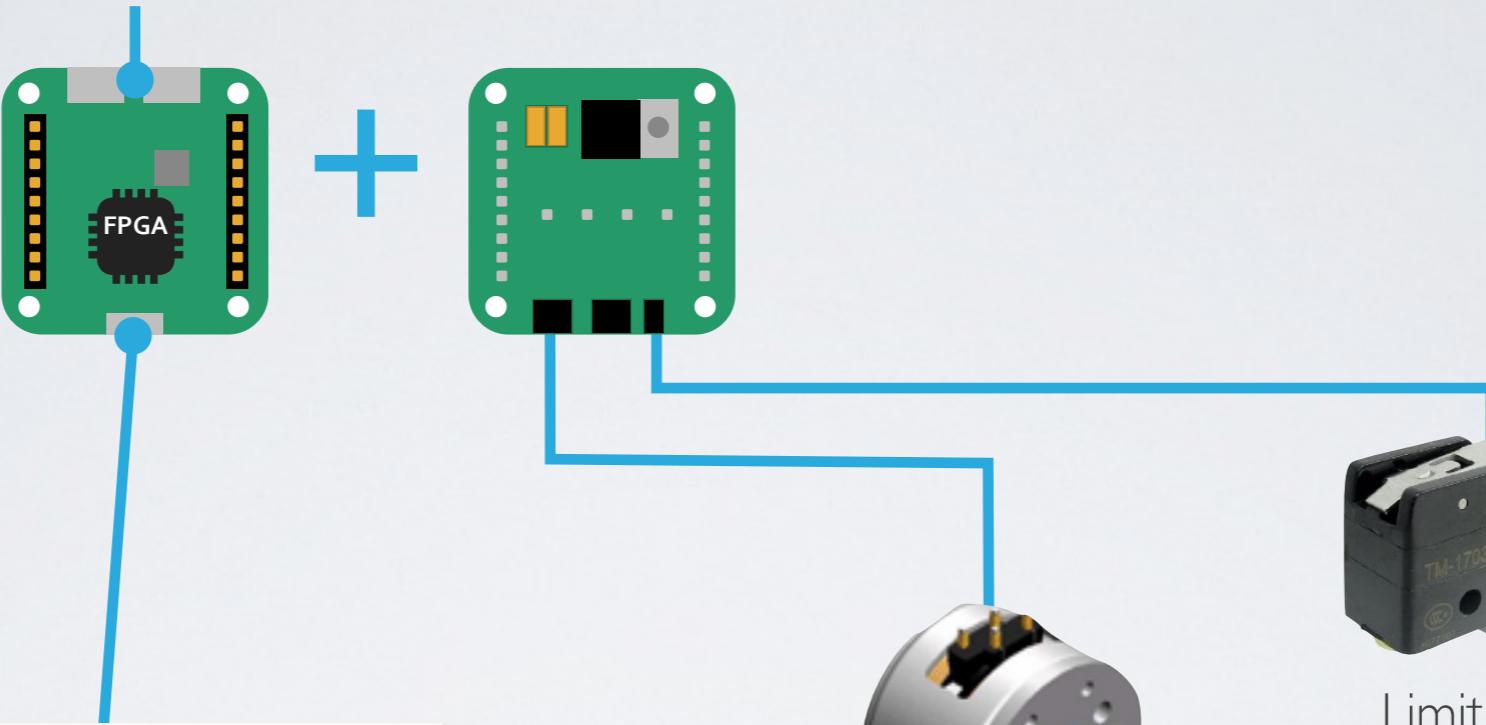
Generic FPGA Board



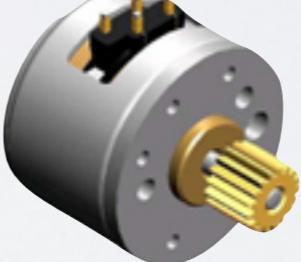


Stepper Motor

Kart i²c interconnect bus



Limit switch



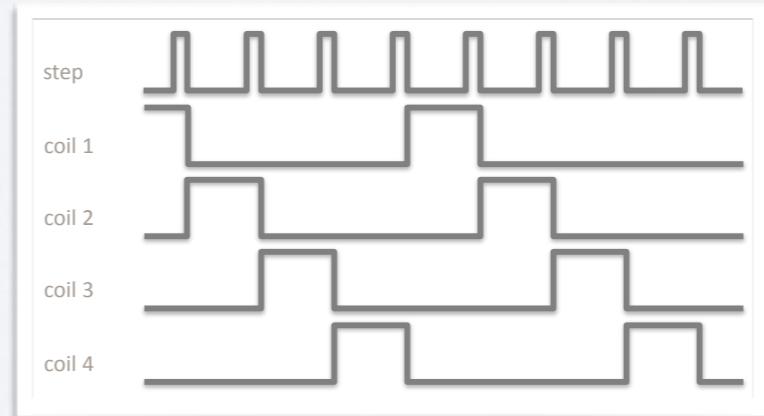
Stepper motor

```

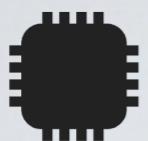
1  -- Company: Young Embedded Systems LLC
2  -- Engineer: Gene Brennen
3  -- Module Name: ARM_SEQ_RAM - Behavioral
4  -- Revisions:
5  -- 0.01 - 07/02/2007 File Created
6  -- Additional Comments:
7  --
8  --
9  --
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.STD_LOGIC_UNSIGNED.ALL;
13 use IEEE.STD_LOGIC_UNSIGNED.ALL;
14
15 entity ClkDiv is
16   Port ( InByt : in STD_LOGIC_VECTOR(3 downto 0);      --<<< Seq_CPLD
17          RegSel : in STD_LOGIC_VECTOR(1 downto 0);      --<<< Seq_CPLD
18          RegScr : in STD_LOGIC;                         --<<< Seq_CPLD
19          MClk : in STD_LOGIC;                          --<<< OSC
20          SeqReset : in STD_LOGIC;                      --<<< Power Monitor
21          ADC_Clk : out STD_LOGIC;                     -->>> ADC
22 end ClkDiv;
23
24 architecture Behavioral of ClkDiv is
25   signal ADC_div : STD_LOGIC_VECTOR(5 downto 0) := "001111";
26   signal ADCClk : STD_LOGIC := '0';
27   signal ClkSel : STD_LOGIC_VECTOR(2 downto 0) := "100";
28
29 begin
30
31   ClkDivP : process(MClk,SeqReset)
32   begin
33     if SeqReset = '0' then
34       if ADCClk <= '0' then
35         ADC_div <= "001001";
36       elsif MClk = '0' and MClk'event then
37         if ADC_div = "000000" then
38           ADCClk <= not(ADCClk);
39         case ClkSel is
40           when "000" =>          -- 20MHz - divide by 2
41             when "001" =>          -- 10MHz
42               ADC_div <= "000001";  -- divide by 4
43             when "010" =>          -- 4MHz
44               ADC_div <= "000100";  -- divide by 10
45             when "011" =>          -- 2MHz
46               ADC_div <= "001001";  -- divide by 20
47             when "100" =>          -- 1MHz
48               ADC_div <= "001001";  -- divide by 40
49             when others =>        -- 400KHz

```

Stepper motor control

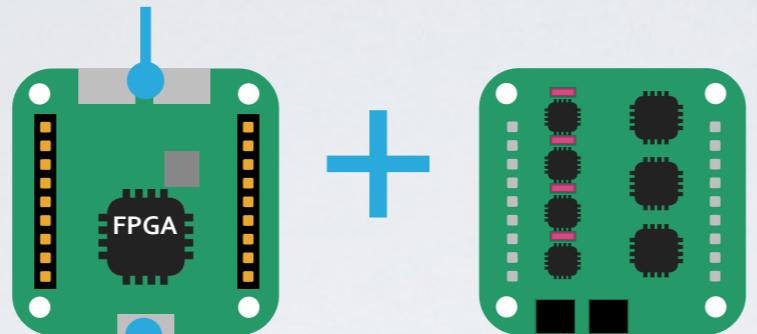


Coil sequence



Drive Motor

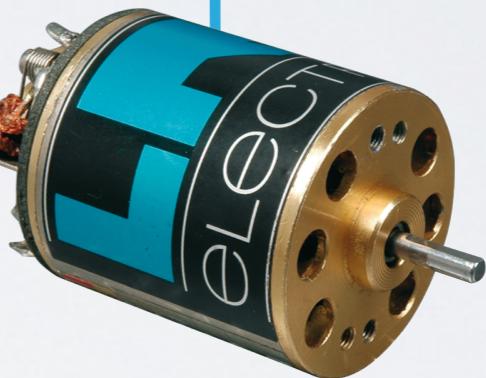
Kart i²c interconnect bus



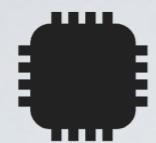
```
1  -- Company: Young Embedded Systems LLC
2  -- Engineer: Gene Brennen
3  -- Module Name: ARM_SEQ_RAM - Behavioral
4  -- Revisions:
5  -- 0.01 - 07/02/2007 File Created
6  -- Additional Comments:
7  --
8  --
9  library IEEE;
10 use IEEE.STD.LOGIC_1164.ALL;
11 use IEEE.STD.LOGIC_ARITH.ALL;
12 use IEEE.STD.LOGIC_UNSIGNED.ALL;
13
14 entity ClkDiv is
15   port ( InByte : in STD_LOGIC_VECTOR(3 downto 0);      --<<-- Seq_CPLD
16         RegSel : in STD_LOGIC_VECTOR(1 downto 0);      --<<-- Seq_CPLD
17         RegStrb : in STD_LOGIC;                         --<<-- Seq_CPLD
18         MClk : in STD_LOGIC;                           --<<-- OSC
19         SeqReset : in STD_LOGIC;                      --<<-- Power Monitor
20         ADC_Clk : out STD_LOGIC);                     -->> ADC
21 end ClkDiv;
22
23 architecture Behavioral of ClkDiv is
24   signal ADC_div : STD_LOGIC_VECTOR(5 downto 0) := "001111";
25   signal ADCClk : STD_LOGIC := '0';
26   signal ClkSel : STD_LOGIC_VECTOR(2 downto 0) := "100";
27
28 begin
29
30   ClkDirP : process(MClk,SeqReset)
31   begin
32     if SeqReset = '0' then
33       if ADCClk <= '0' then
34         ADC_div <= "001001";
35       else
36         elsif MClk = '0' and MClk'event then
37           if ADC_div = "000000" then
38             ADCClk <= not(ADCClk);
39             case ClkSel is
40               when "000" =>          -- 20MHz - divide by 2
41                 when "001" =>          -- 10MHz
42                   ADC_div <= "000001";  -- divide by 4
43                   when "010" =>          -- 4MHz
44                   ADC_div <= "000100";  -- divide by 10
45                   when "011" =>          -- 2MHz
46                   ADC_div <= "001001";  -- divide by 20
47                   when "100" =>          -- 1MHz
48                   ADC_div <= "001001";  -- divide by 40
49                   when others =>        -- 400KHz

```

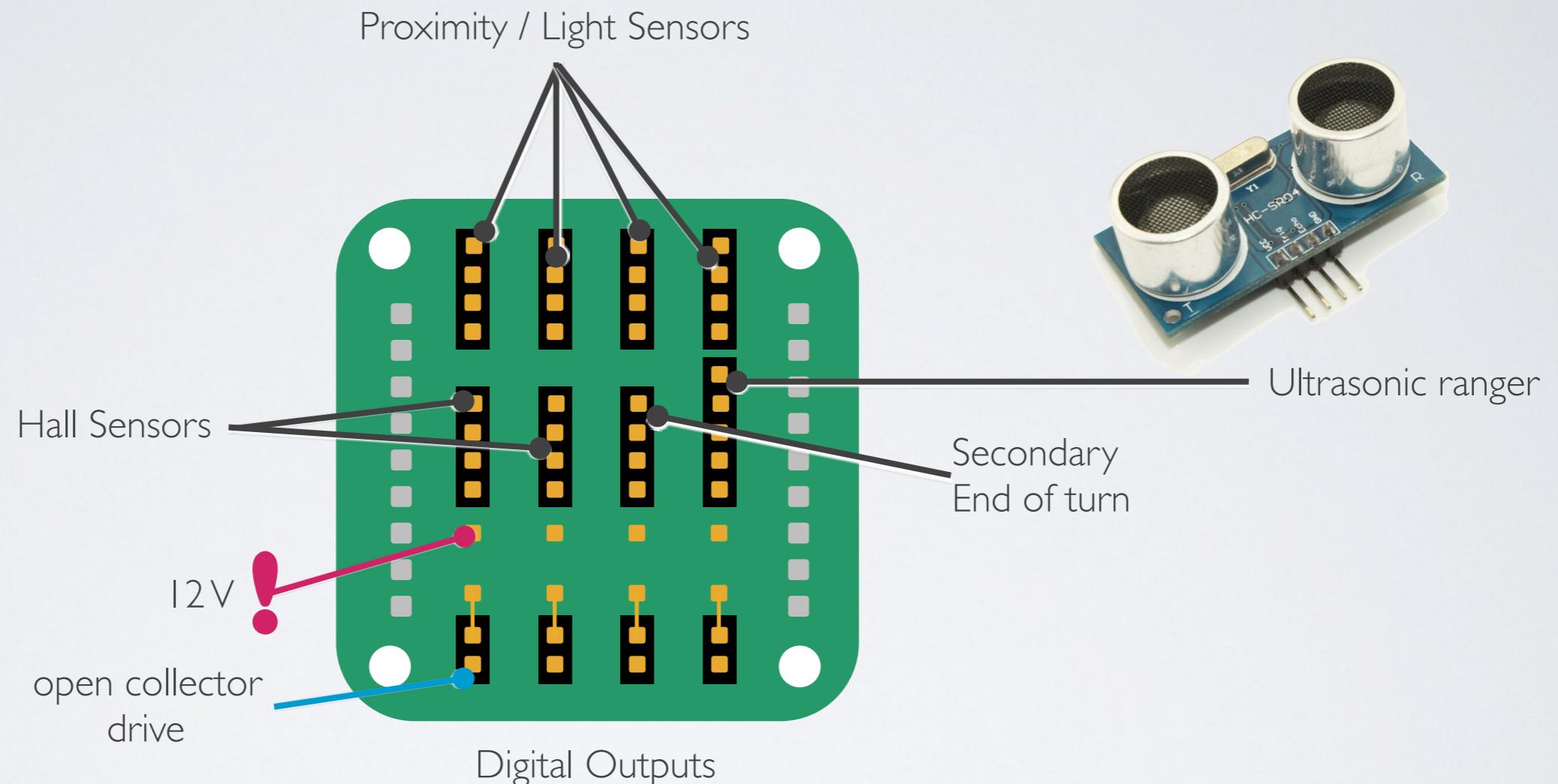
Drive PWM control

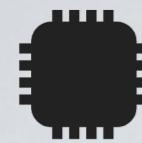


Drive motor

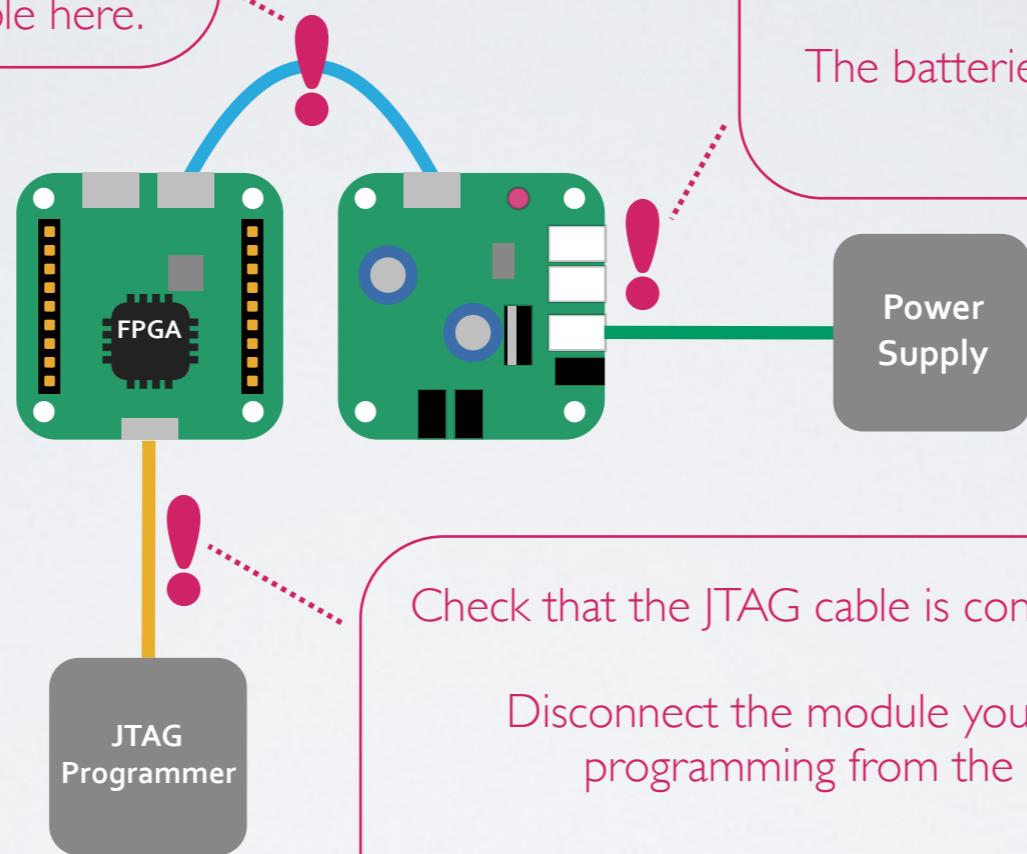


IO Board





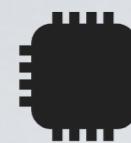
Avoid Hardware Damages



If you connect something wrong, the FPGA might be **damaged**.

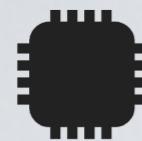
The costs to change a FPGA are about **50 SFr**.

You will be charged for the reparation if you did not follow this guidelines!



Hardware Goals

- Control block for DC motor
 - Pulse Width Modulation (PWM) generator
- Control block for stepper motor
 - 4 Coil forward/backward sequence generator
- Hall-Sensor Counter
- Various additional sensors and actuators
 - Personal ideas are welcome



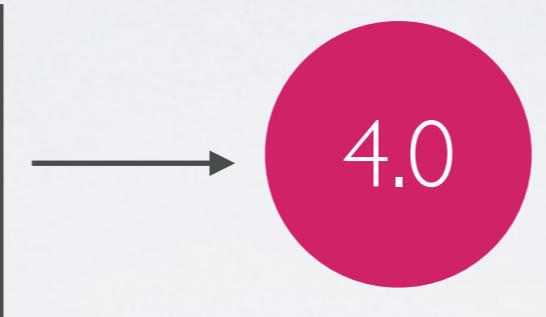
Hardware Grade



Presentation of blocks and simulation results during morning of the last day

All mandatory features

Direction Stepper control
Speed PWM control
Hall sensor counter



Per optional feature

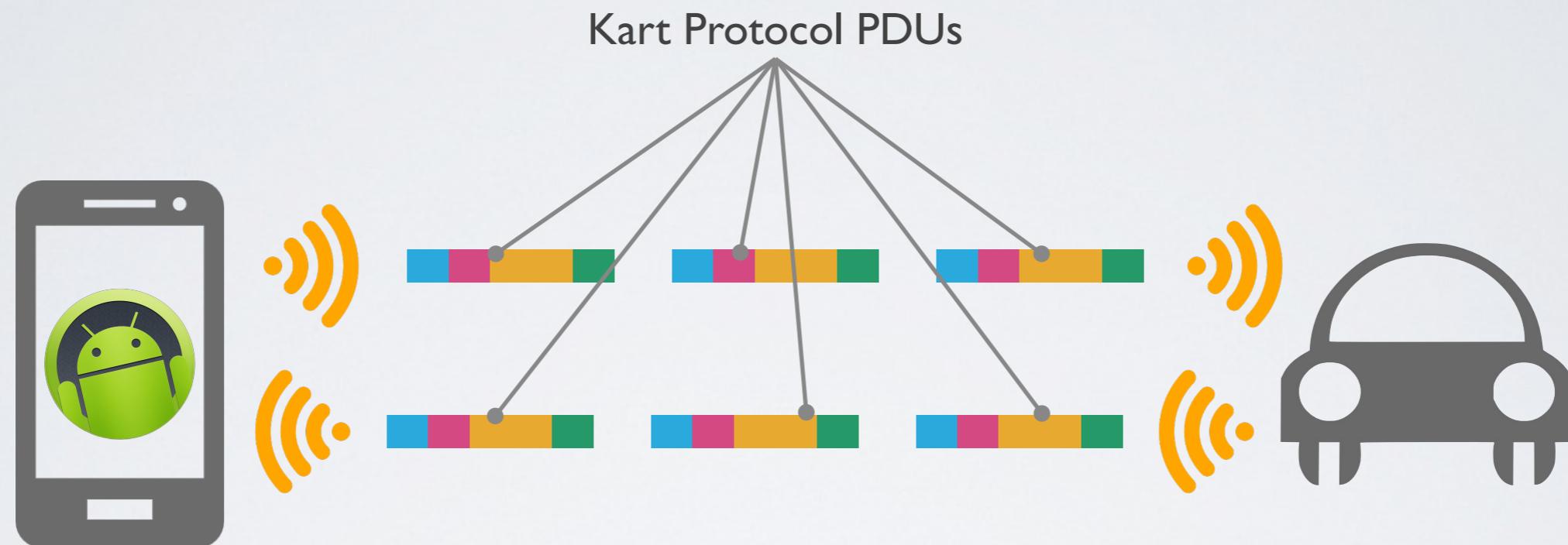
Ultrasound sensor
Emergency Stop (Proximity sensor)
Other improvements



INF part

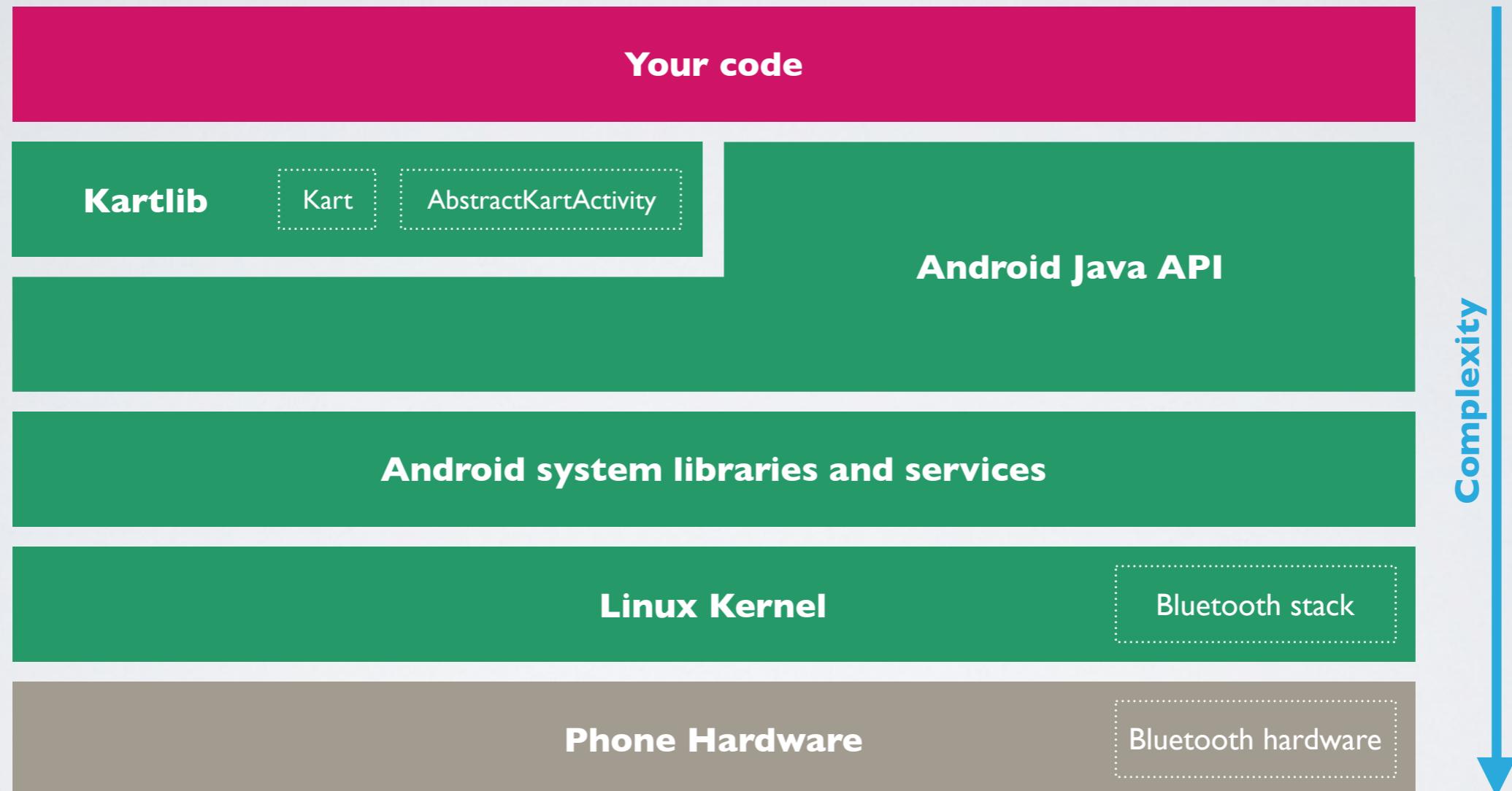


Remote Control Protocol





Remote Control Protocol





Remote Control Android App

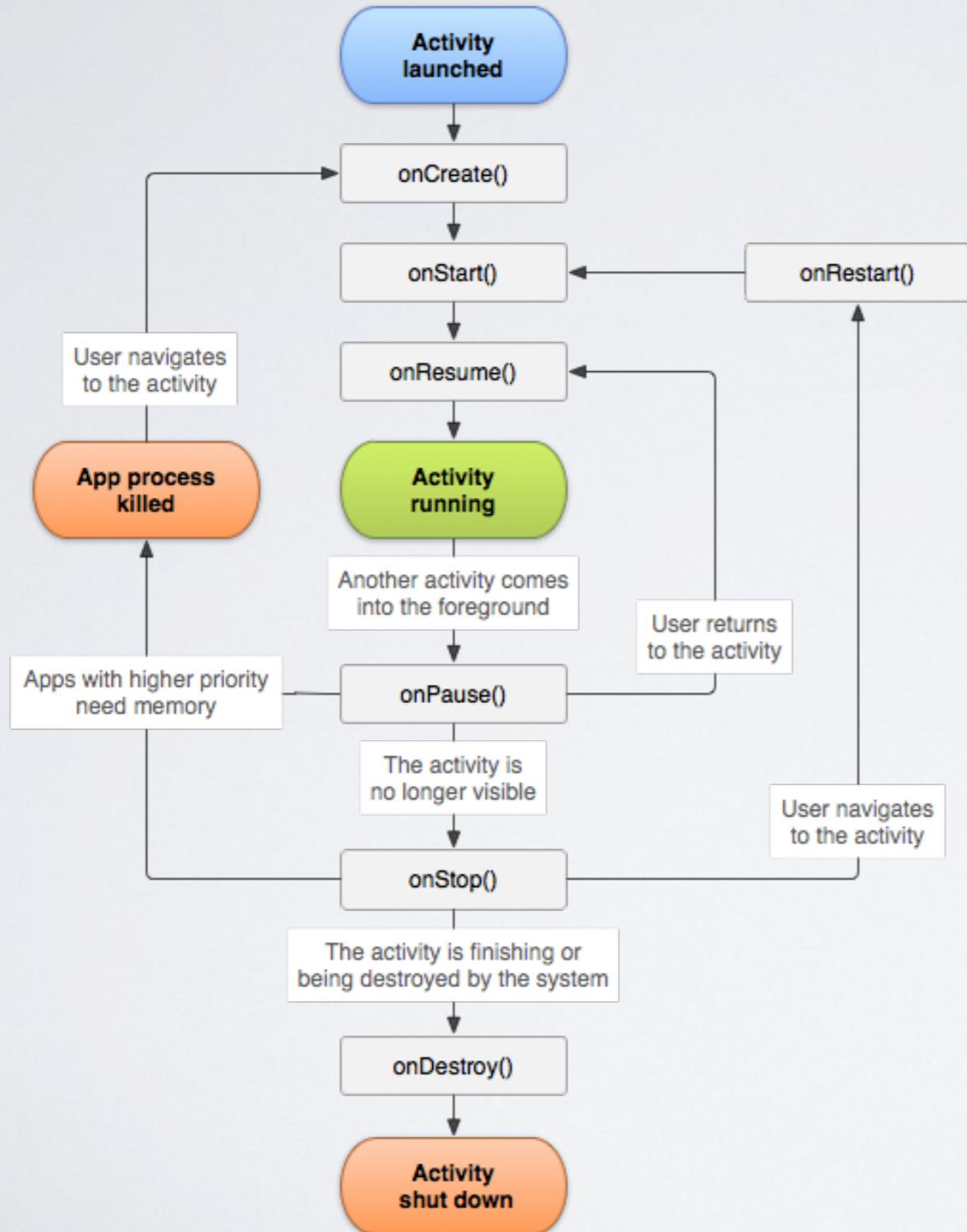




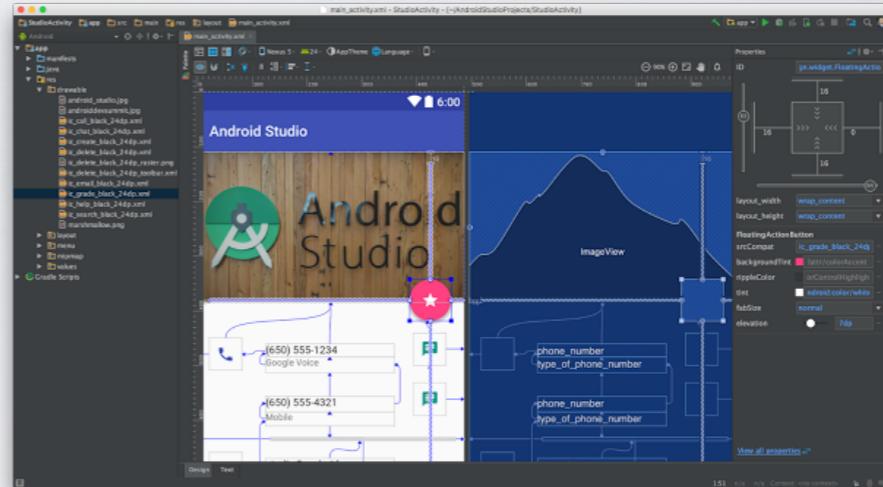
- Mobile **O**perating **S**ystem developed by **Alphabet** (Google).
- **Abstracts hardware** from different manufacturers to a **common API**
- Applications are written in **Java or Kotlin** and run on a **Virtual Machine** (Dalvik or ART)
- Android is **open source**, based on **Linux**.
- The **SDK & Android Studio** (based on **IntelliJ IDEA**) are free to use and allow everyone to build applications for Android.



Android Application lifecycle



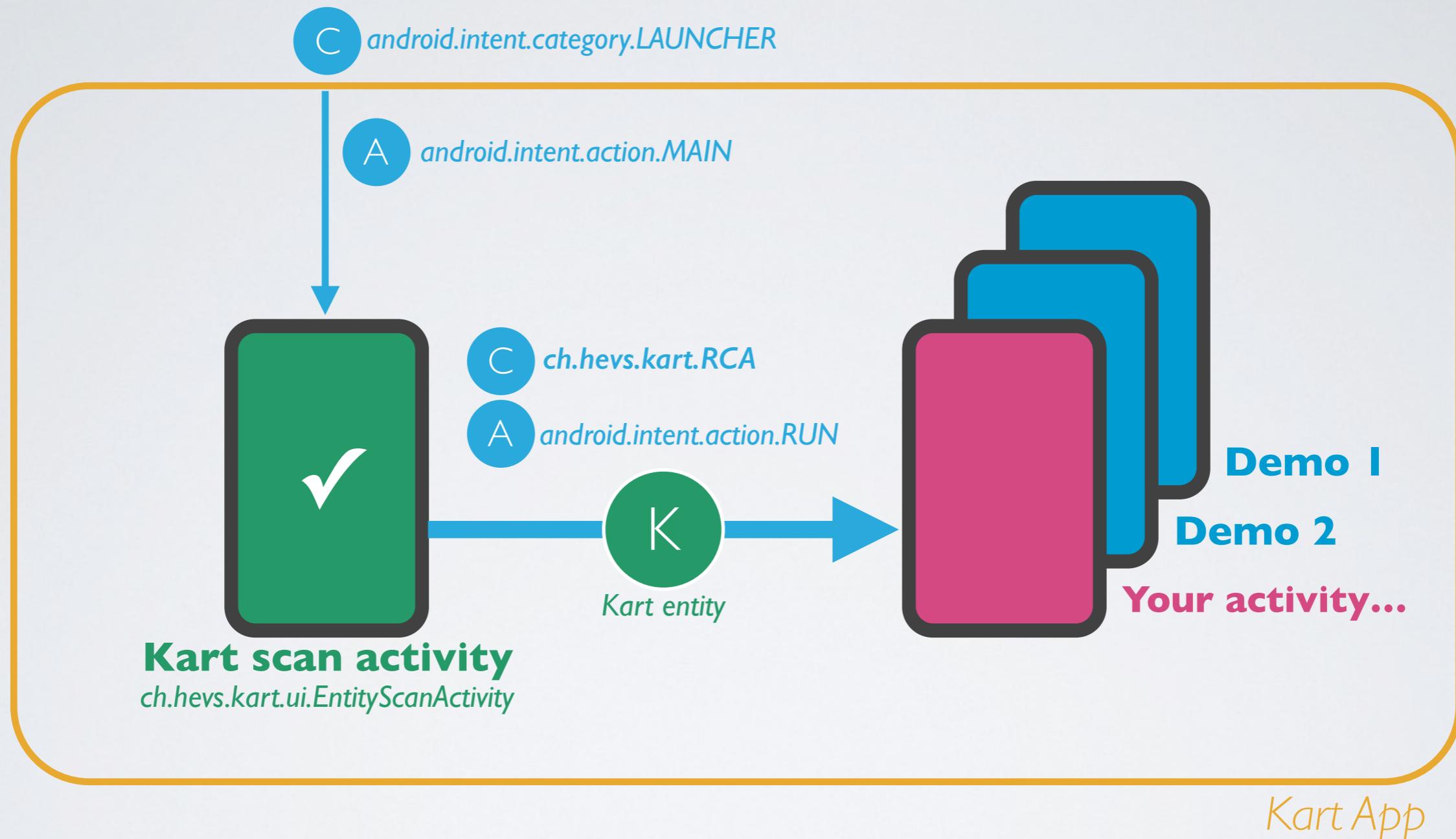
Layouts



- UI layouts can be designed using an editor integrated into Android Studio.
 - Layouts are serialized to XML files.
 - Those Layouts can be loaded in code.

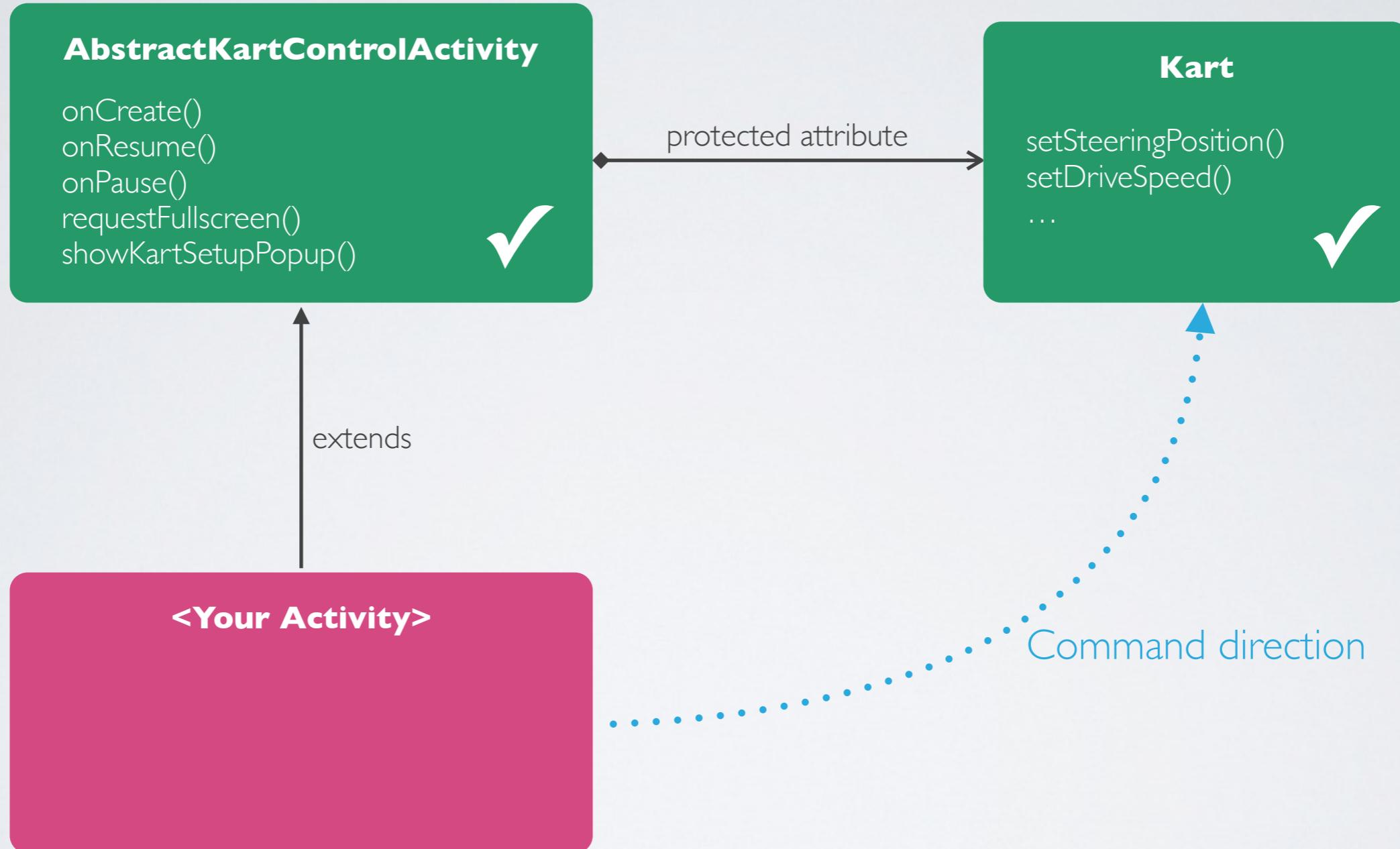


Remote Control Android App



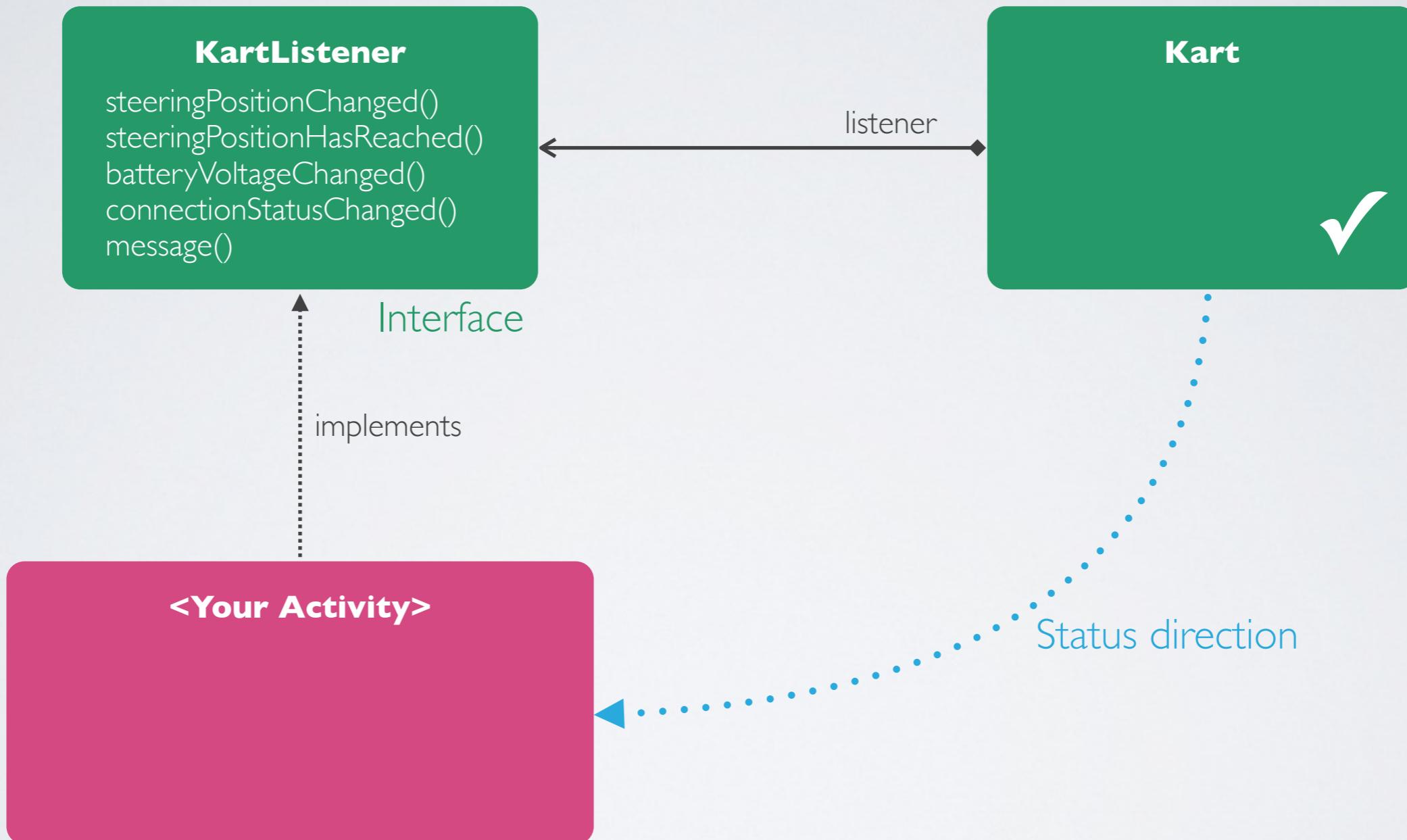


Remote Control Android App



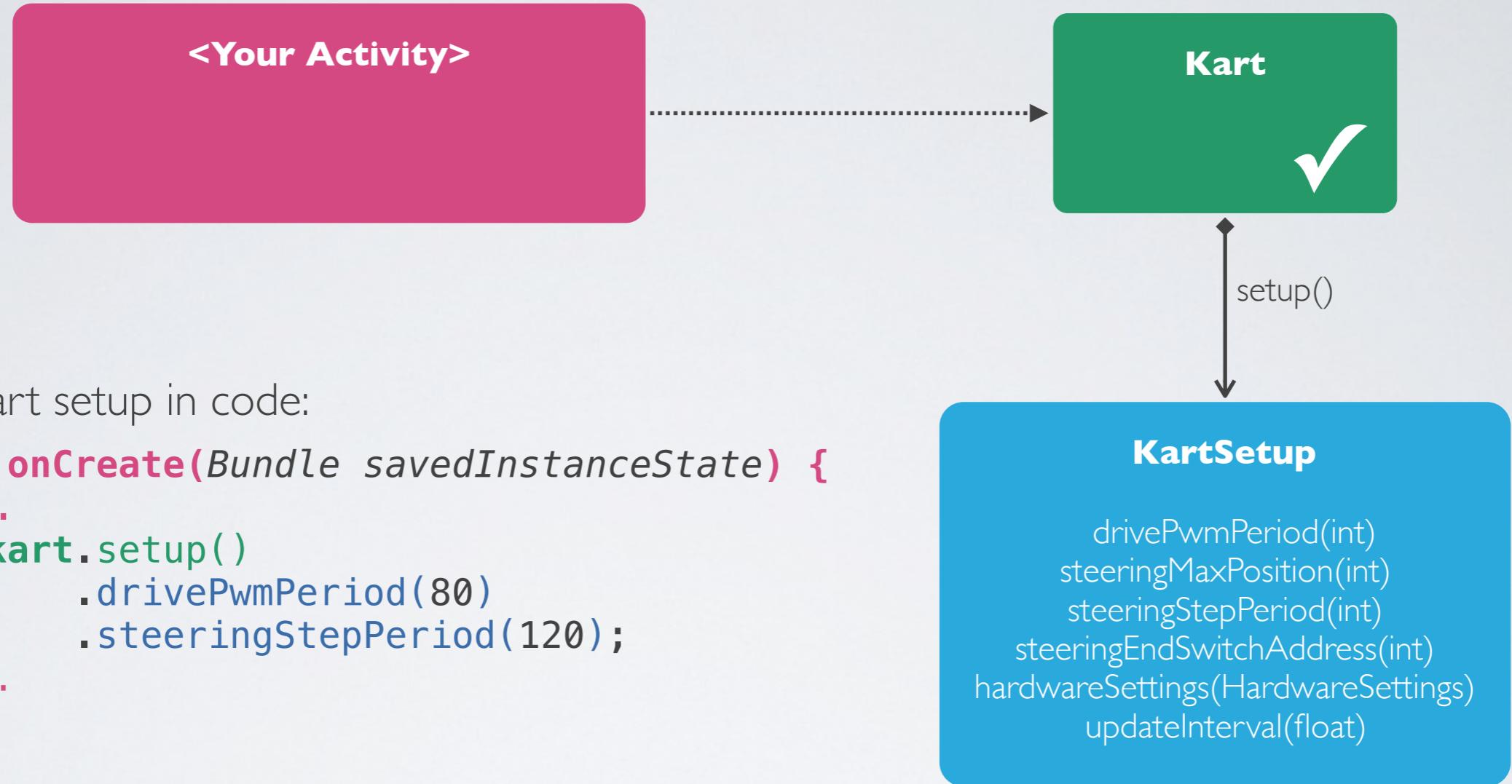


Remote Control Android App





Remote Control Android App



Manual kart setup in code:

```
void onCreate(Bundle savedInstanceState) {  
    ...  
    kart.setup()  
        .drivePwmPeriod(80)  
        .steeringStepPeriod(120);  
    ...  
}
```

Using the provided kart setup UI:

```
void onCreate(Bundle savedInstanceState) {  
    ...  
    showKartSetupPopup();  
    ...  
}
```



Software Goals

- **Slider control**
 - Direction
 - Speed
- **Progress Bar status**
 - Battery level
 - Steering position
- **Accelerometer (Orientation) control**
 - Button to enable orientation control
 - Device orientation controls sliders or kart



Software Grade



Functional blackbox tests
during morning of the last
day

All mandatory features

Direction control
Speed control
Battery display
Direction display

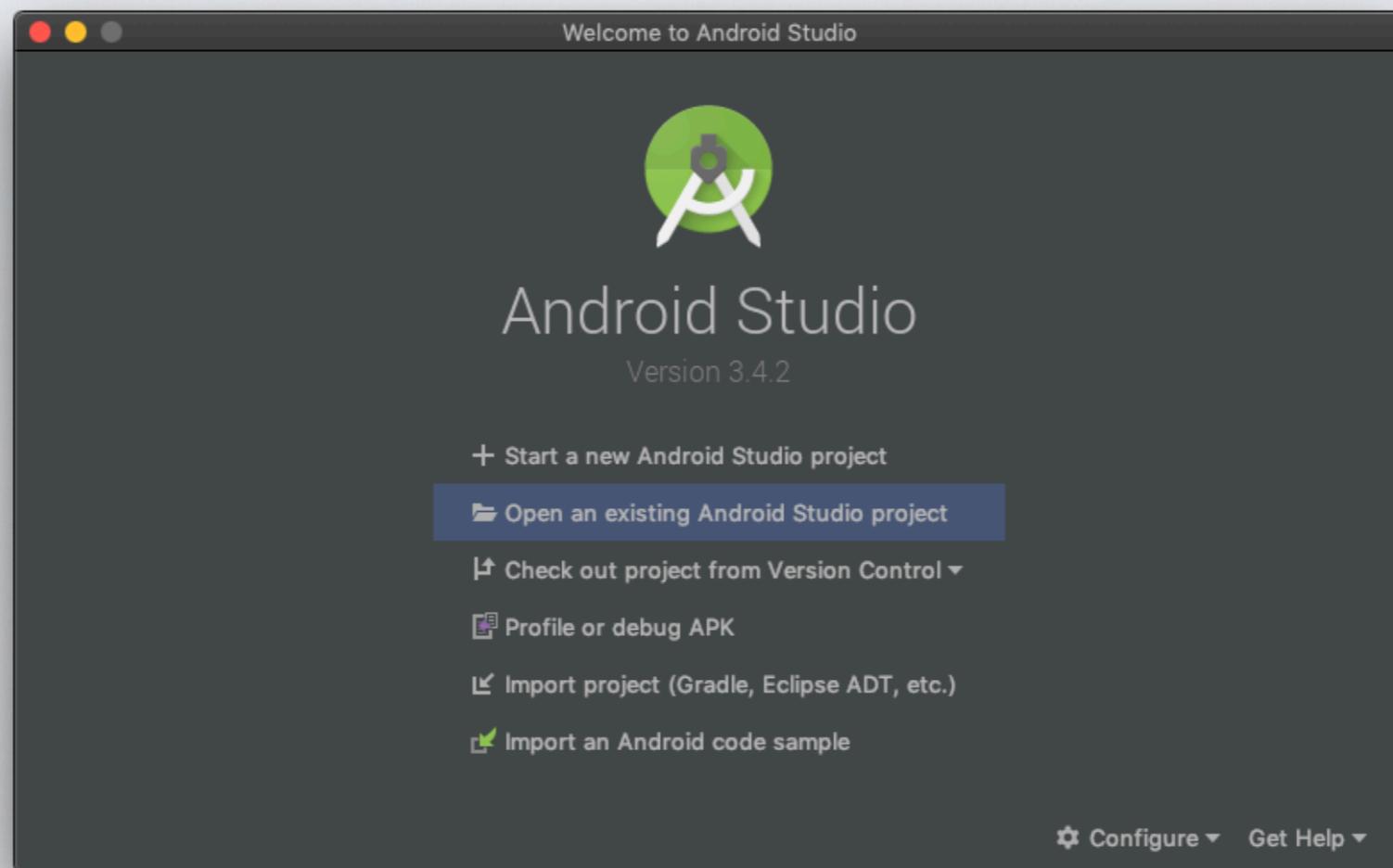


Per optional feature

LED's
Ultrasound sensor
Proximity sensor
Hall sensor
Steering assistant
Other improvements



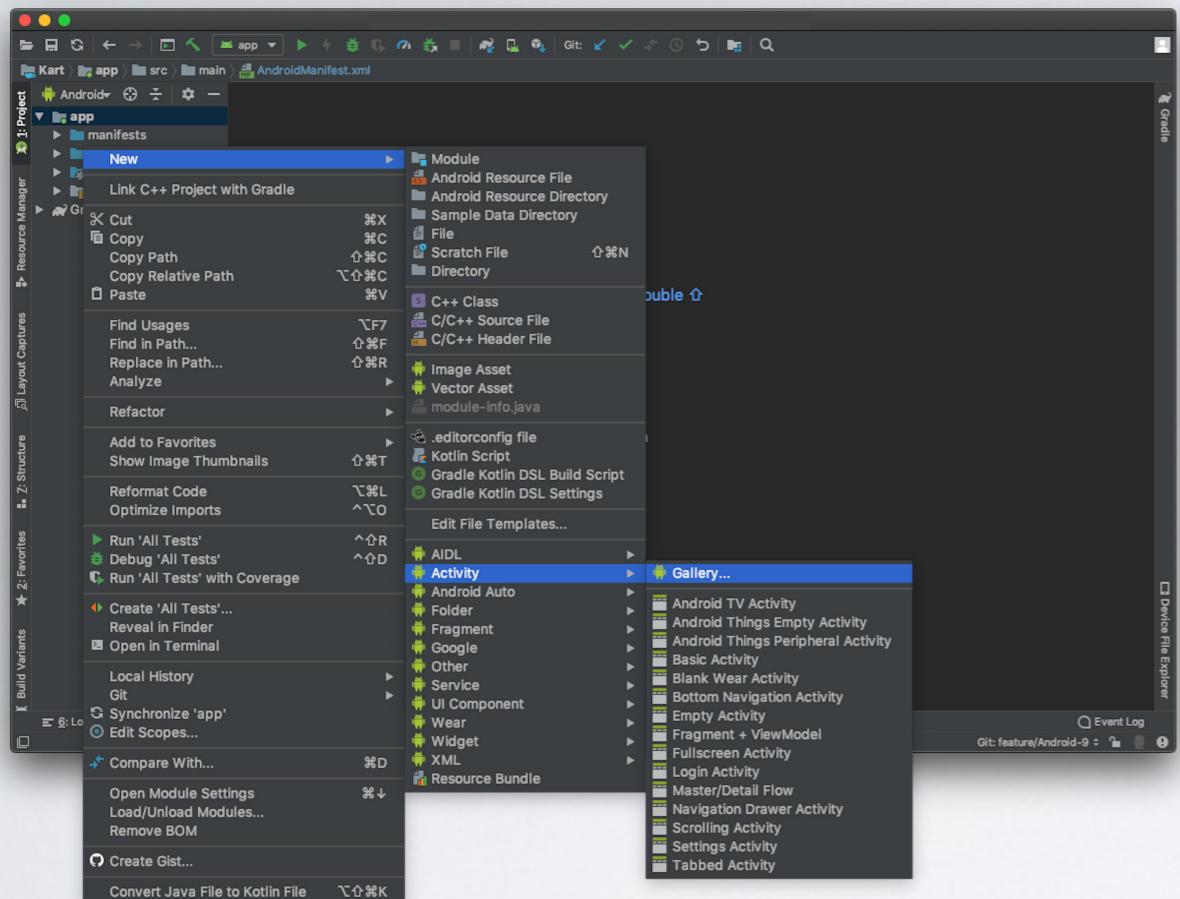
- ① **Download** the Kart project template from the wiki
- ② **Extract** the archive to the **local** disk
- ③ **Open** the extracted folder in Android Studio



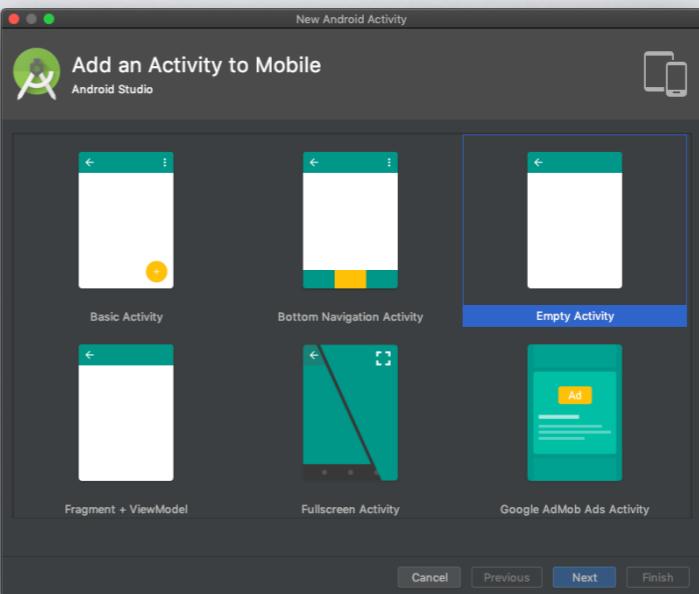


Android Studio

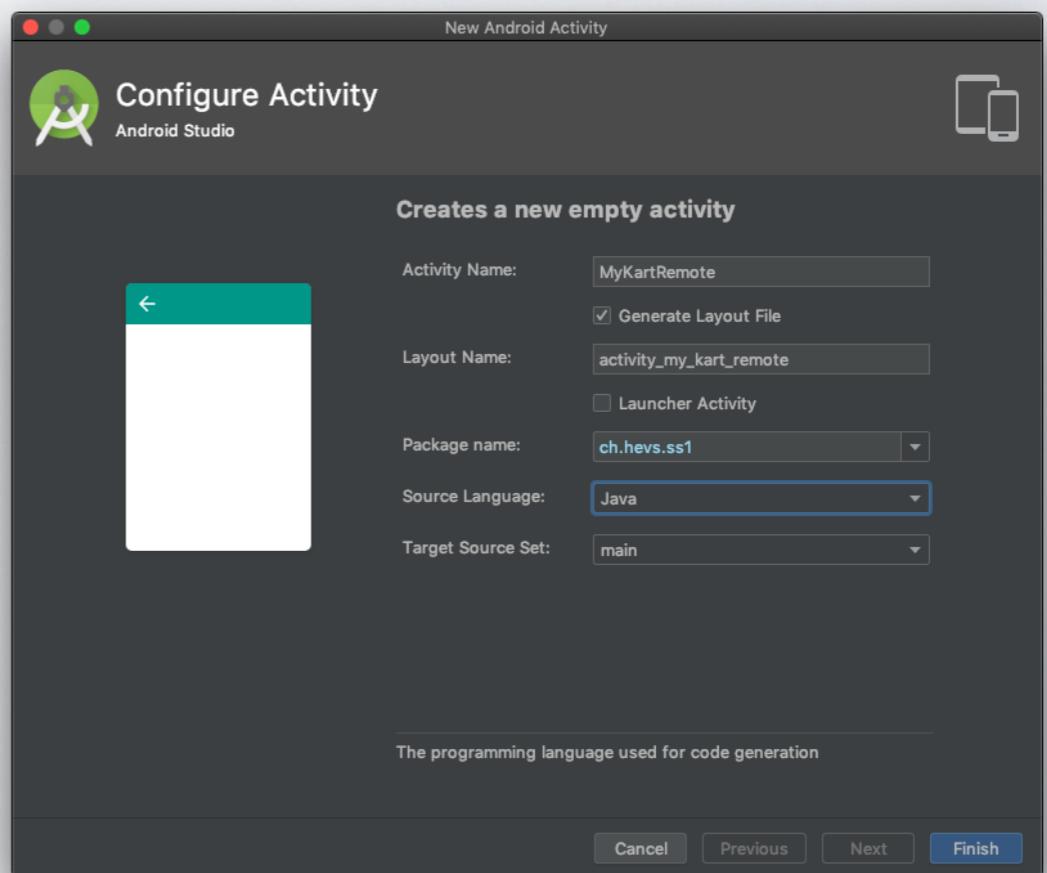
4 Select New > Activity > Gallery...



5 Select Empty Activity



6 Configure Activity





Android Studio

7 Modify Activity Java code

```
package ch.hevs.ss1;

import android.os.Bundle;

import ch.hevs.kart.AbstractKartControlActivity;
import ch.hevs.kart.Kart;

public class MyKartRemote extends AbstractKartControlActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my_kart_remote);
    }

    @Override
    public void steeringPositionChanged(Kart kart, int i, float v) { }

    @Override
    public void steeringPositionReachedChanged(Kart kart, boolean b) { }

    @Override
    public void batteryVoltageChanged(Kart kart, float v) { }

    @Override
    public void sequenceCompleted(Kart kart) { }

    @Override
    public void connectionStatusChanged(Kart kart, boolean b) { }

    @Override
    public void message(Kart kart, String s) { }
}
```

8 Modify AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="ch.hevs.ss1">

    <application
        android:allowBackup="false"
        android:label="@string/app_name"
        android:supportsRtl="true"
        tools:ignore="GoogleAppIndexingWarning">

        <activity android:name=".MyKartRemote">
            <intent-filter>
                <action android:name="android.intent.action.RUN" />
                <category android:name="ch.hevs.kart.RCA" />
            </intent-filter>
        </activity>
    </application>

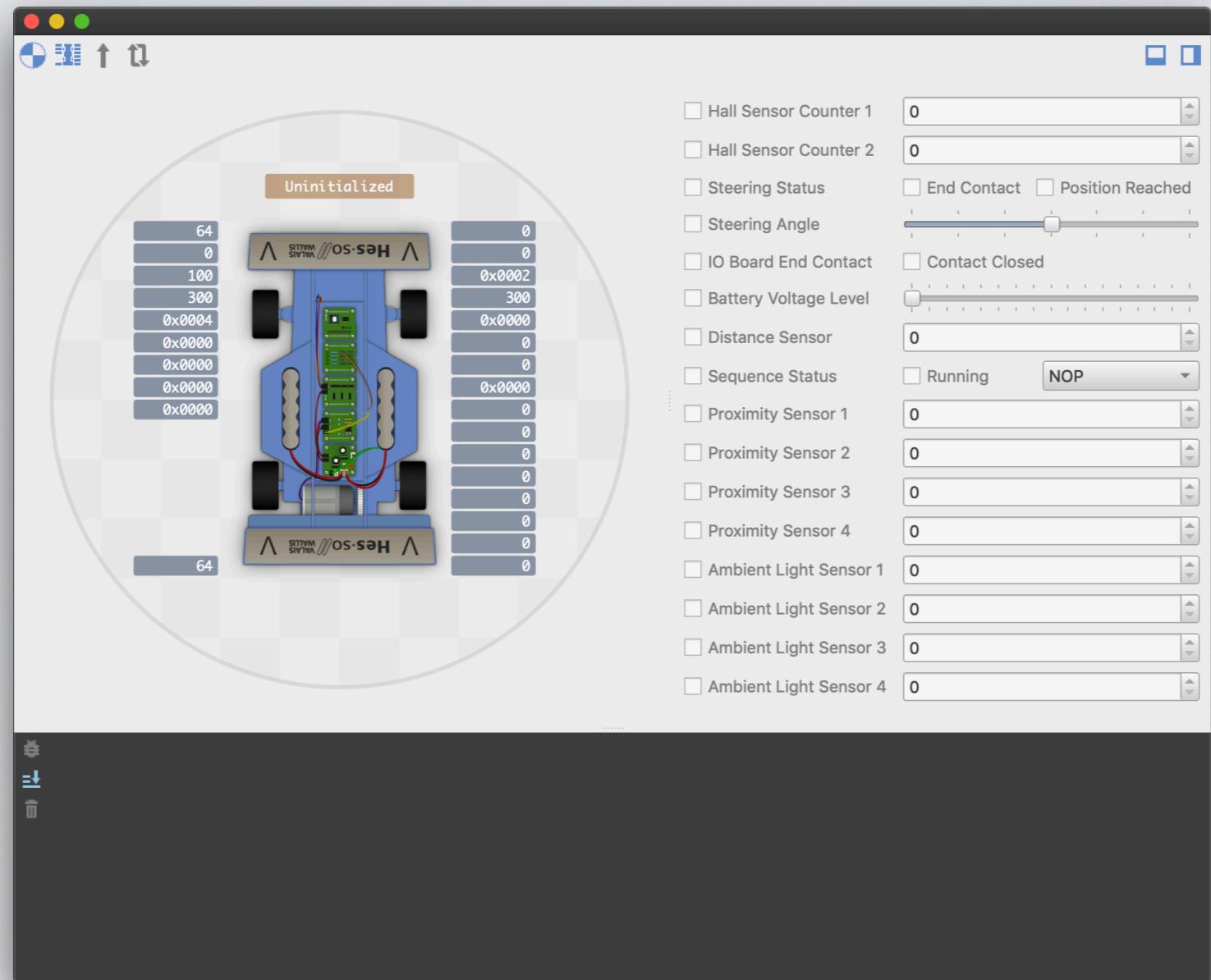
</manifest>
```

- Connect Phone to PC's USB port
- Power on the phone
- Use default configurations during setup wizard
- Enable developer mode:
Go to Settings > System > About Phone and press 7 times on „Build Number“
- Enable USB debugging:
Go to Settings > System > Developer options and check „USB debugging“
- Install and start your Android application:
 - In Android Studio, press green  button and select honor 10 Lite phone from the list.
 - On the phone, answer yes to allow USB debugging in popup.
 - Now your application should be running on the phone.





Virtual Kart





Download **Kart.apk** from:

<http://wiki.hevs.ch/fsi/index.php5/Kart>

Install it using the **adb** command line tool:

```
c:\Users\your.account> c:\Android\sdk\platform-tools\adb install Downloads\Kart.apk
3979 KB/s (937315 bytes in 0.230s)
    pkg: /data/local/tmp/Kart.apk
```

Success



Tipp #1: Read the docs



- You find all information needed here:
 - Your copy of the kart project documentation and tasks document
 - Kart wiki: <http://wiki.hevs.ch/fsi/index.php5/Kart>
 - Kart project JavaDoc: <http://kart-javadoc.hevs.ch>
 - Android: <https://developer.android.com/index.html>

Ask us, we kindly like to help you!



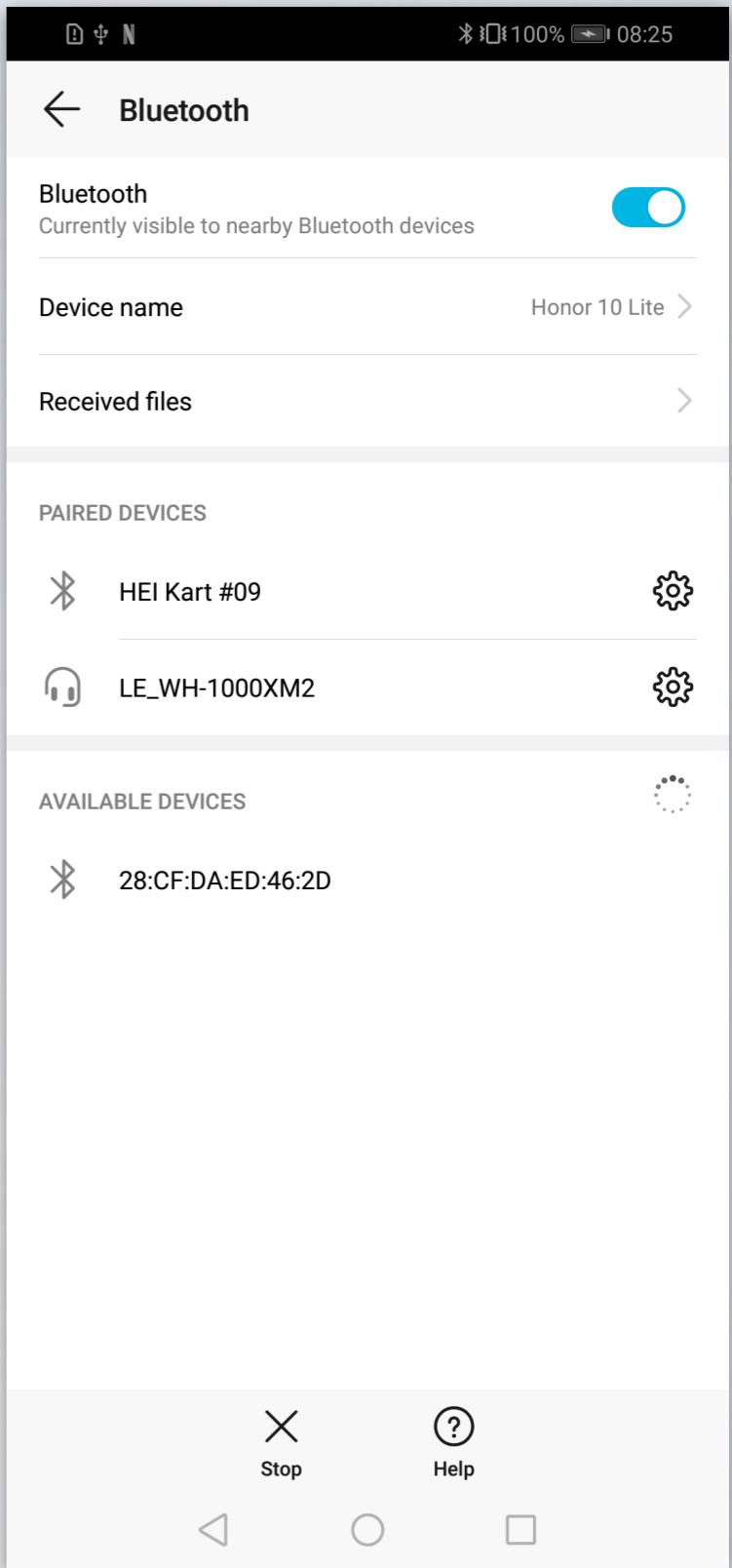
Tipp #2: Pair with your kart

You need to pair with your kart before it will be listed by the Kart app

Open “Settings”

Go to “Device connectivity” > “Bluetooth”

Select your kart in “Available Devices”





Tipp #3: No infinite loops

```
// Blink a led
try {
    while (true) {
        kart.setLed(0, true);
        Thread.sleep(500);
        kart.setLed(0, false);
        Thread.sleep(500);
    }
} catch (InterruptedException e) {
    e.printStackTrace();
}
```

If you add infinite loops to the main thread, your application will crash!





Tipp #4: Periodic Timer

Inside your Activity, add the attribute:

```
private Timer ledBlinker = new Timer() {  
    @Override  
    public void onTimeout() {  
        kart.toggleLed(0);  
    }  
};
```

You can start the timer using:

```
ledBlinker.schedulePeriodically(500);
```

You can stop the timer with:

```
ledBlinker.stop();
```

Documentation:

<http://kart-javadoc.hevs.ch/ch/hevs/kart/utils/Timer.html>



Tipp #5: Doing something later

```
Timer doLater = new Timer() {  
    @Override  
    public void onTimeout() {  
        kart.setLed(0, true);  
    }  
};  
doLater.scheduleOnce(5000);
```

This code snippet will turn the LED 0 on
after 5 seconds.

Documentation:

<http://kart-javadoc.hevs.ch/ch/hevs/kart/utils/Timer.html>



Tipp #6: Animations...

```
Animation animation = Animation.Builder(kart)
    .ledOn(0).ledOff(1).wait(100)
    .ledOff(0).ledOn(1).wait(100)
    .build();
animation.loop();
```

The animation will turn LED 0 on and LED 1 off, then wait for 0.1s. Next it will turn LED 0 off and LED 1 on and then wait again for 0.1s. The animation is looped until the method **cancel()** is called...

Documentation:

<http://kart-javadoc.hevs.ch/ch/hevs/kart/utils/Animation.html>

<http://kart-javadoc.hevs.ch/ch/hevs/kart/utils/Animation.Builder.html>



Tipp #7: Enumerations

You may not have seen Java Enumerations during your first year informatics course...

```
public enum DayOfWeek {  
    MONDAY,  
    TUESDAY,  
    WEDNESDAY,  
    THURSDAY,  
    FRIDAY,  
    SATURDAY,  
    SUNDAY  
}
```

DayofWeek.java

```
Worktime  
Today is TUESDAY  
MONDAY  
TUESDAY  
WEDNESDAY  
THURSDAY  
FRIDAY  
SATURDAY  
SUNDAY
```

Program output

```
DayOfWeek today = DayOfWeek.TUESDAY;  
  
if (today == DayOfWeek.MONDAY) {  
    System.out.println("Today sucks!");  
}  
  
switch (today) {  
    case SATURDAY:  
    case SUNDAY:  
        System.out.println("Freetime");  
  
    default:  
        System.out.println("Worktime");  
}  
  
System.out.println("Today is " + today);  
  
for (DayOfWeek dow: DayOfWeek.values()) {  
    System.out.println(dow);  
}
```

Main.java



Tipp #7: Enumerations

```
public enum KartControlRegister {  
    DrivePwmPeriod(0),  
    DriveSpeed(1),  
    SteeringStepPeriod(2),  
    SteeringPosition(3),  
    SteeringEndSwitchAddress(4),  
    HardwareSettings(5),  
    LEDs(6),  
    SequenceControlRegister(7),  
    SequenceFifoRegister(8),  
    UpdateInterval(15);  
  
    private final int address;  
  
    KartControlRegister(final int address) {  
        this.address = address;  
    }  
  
    public int getAddress() {  
        return address;  
    }  
  
    static final int COUNT = 16;  
}
```

KartControlRegister.java

KartControlRegister has 16 registers.
LEDs register has address 6

Program output

```
System.out.println("KartControlRegister has " + KartControlRegister.COUNT + " registers.");  
  
KartControlRegister register = KartControlRegister.LEDs;  
System.out.println(register + " register has address " + register.getAddress());
```

Main.java