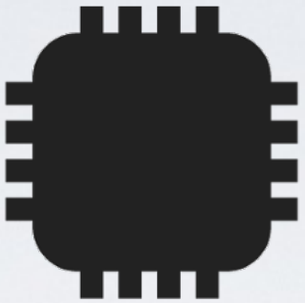
Kart Programming

François Corthay | Oliver Gubler

Michael Clausen | Lucas Bonvin

Goals



Use basic elements seen in ELN course.

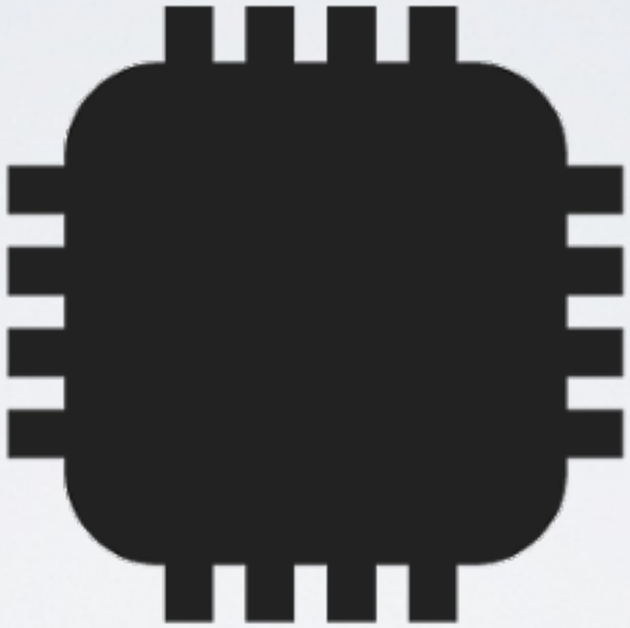
Use basic elements seen in INF I course.

Goals

Galaxy Nexus

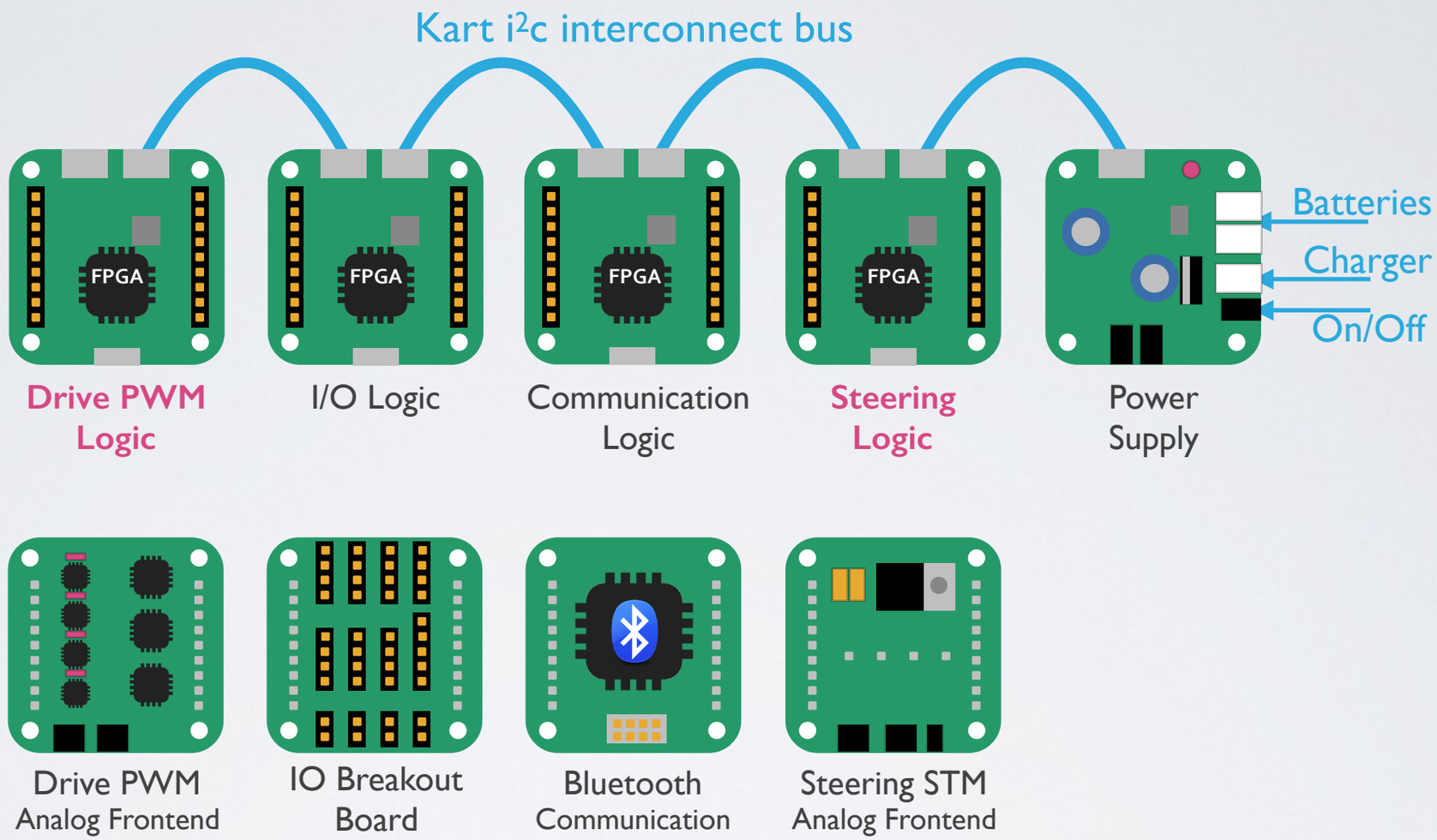
Bluetooth

Kart

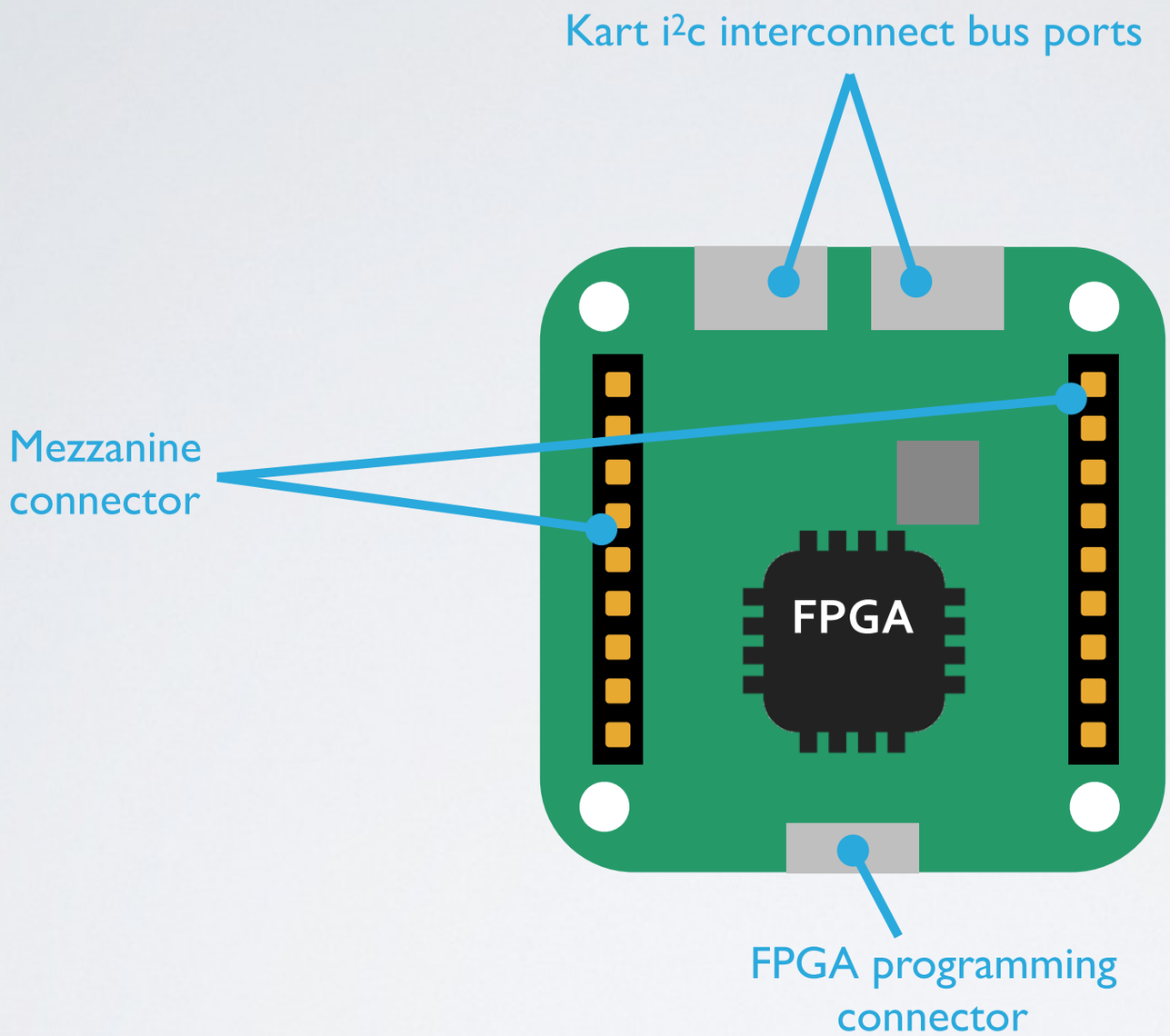


ELN part

Modular concept



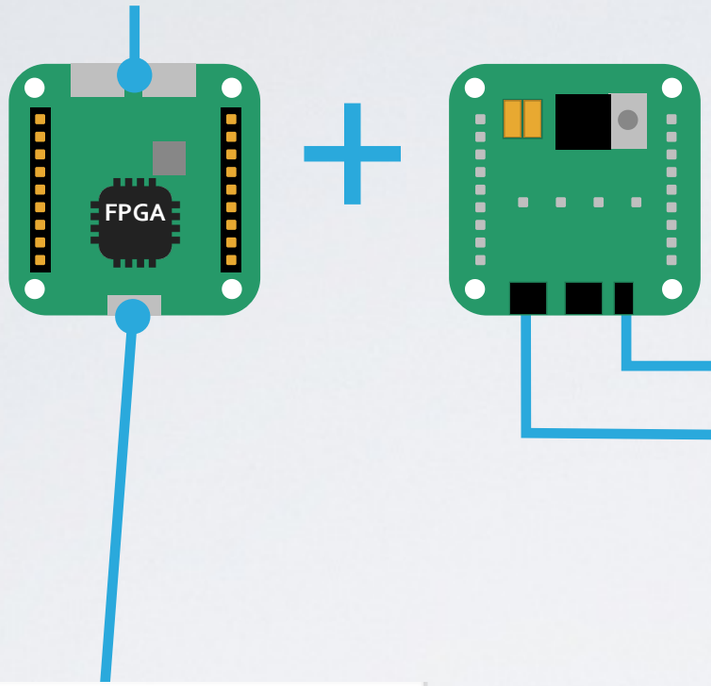
Generic FPGA Board

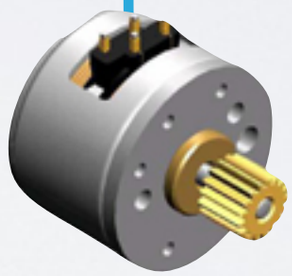


Stepper Motor

Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Kart i²c interconnect bus



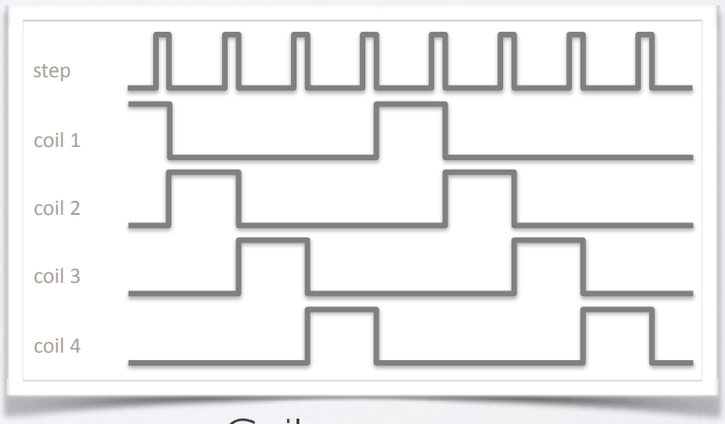


Stepper motor

Limit switch

```
--------------------------------------------------------------------------------
-- Company: Young Embedded Systems LLC
-- Engineer: Gene Breniman
-- Module Name:    ARM_SEQ_RAM- - Behavioral
-- Revisions:
--    0.01 - 07/02/2007 File Created
-- Additional Comments:
--
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ClkDiv is
    Port (  InByte : in STD_LOGIC_VECTOR(3 downto 0);       --<-- Seq_CPLD
            RegSel : in STD_LOGIC_VECTOR(1 downto 0);       --<-- Seq_CPLD
            RegStrb : in STD_LOGIC;                         --<-- Seq_CPLD
            MClk : in  STD_LOGIC;                           --<-- OSC
            SeqReset : in STD_LOGIC;                        --<-- Power Monitor
            ADC_Clk : out STD_LOGIC);                       -->-- ADC
end ClkDiv;

architecture Behavioral of ClkDiv is
   signal   ADC_div :      STD_LOGIC_VECTOR(5 downto 0) := "001111";
   signal   ADCClk :       STD_LOGIC := '0';
   signal   ClkSel :       STD_LOGIC_VECTOR(2 downto 0) := "100";

begin

   ClkDivP : process(Mclk,SeqReset)
      begin
      if SeqReset = '0' then
         ADCClk <= '0';
         ADC_div <= "001001";
      elsif Mclk = '0' and Mclk'event then
         if ADC_div = "000000" then
            ADCClk <= not(ADCClk);
            case ClkSel is
               when "000" =>                    -- 20MHz - divide by 2
               when "001" =>                    -- 10MHz
                  ADC_div <= "000001";             -- divide by 4
               when "010" =>                    -- 4MHz
                  ADC_div <= "000100";             -- divide by 10
               when "011" =>                    -- 2MHz
                  ADC_div <= "001001";             -- divide by 20
               when "100" =>                    -- 1MHz
                  ADC_div <= "001001";             -- divide by 40
               when others =>                   -- 400KHz
```

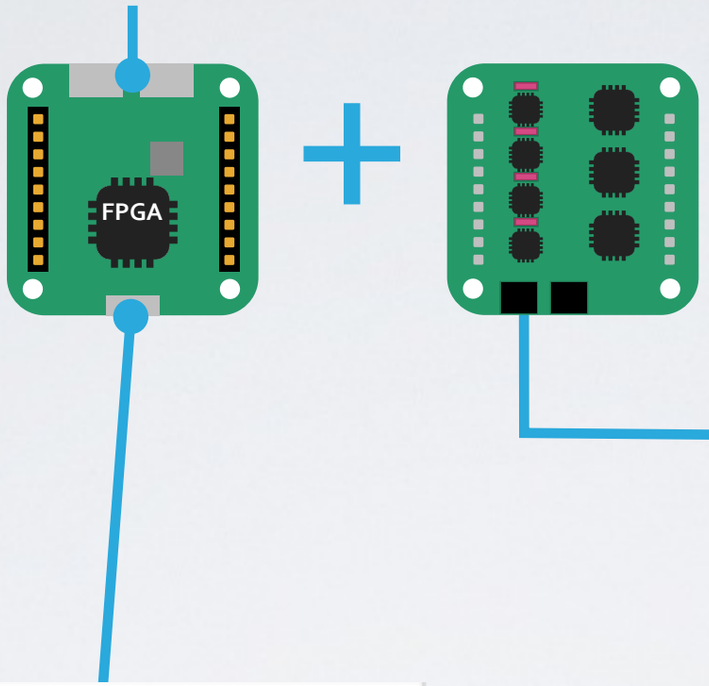
Stepper motor control

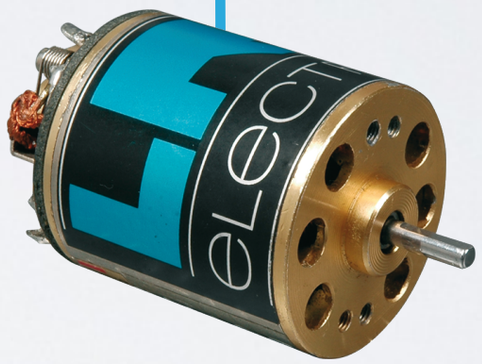


Coil sequence

Drive Motor

Kart i²c interconnect bus



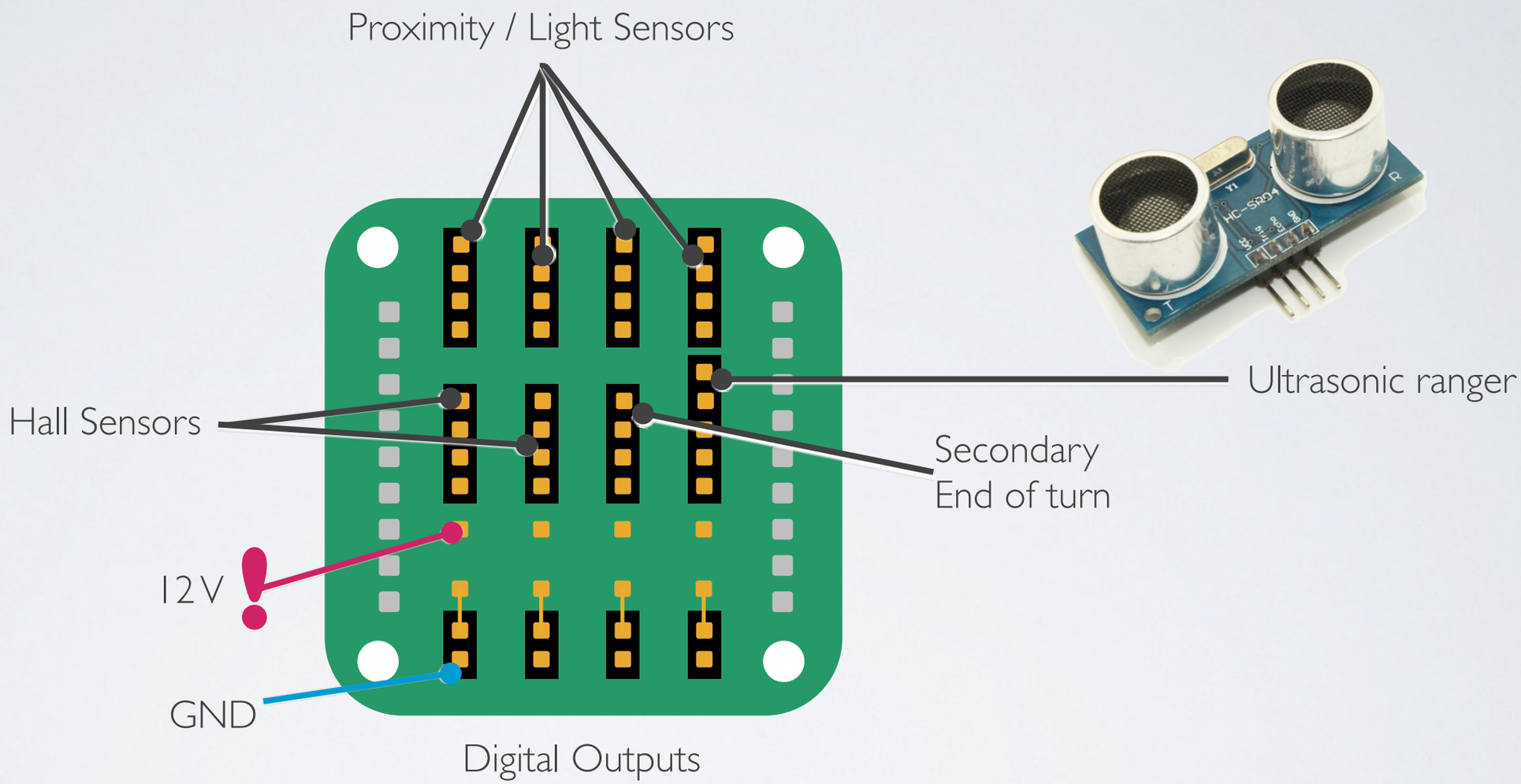


Drive motor

```
1  ----------------------------------------------------------------------------------
2  -- Company: Young Embedded Systems LLC
3  -- Engineer: Gene Breniman
4  -- Module Name:    ARM_SEQ_RAM- - Behavioral
5  -- Revisions:
6  --    0.01 - 07/02/2007 File Created
7  -- Additional Comments:
8  --
9  ----------------------------------------------------------------------------------
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.STD_LOGIC_ARITH.ALL;
13 use IEEE.STD_LOGIC_UNSIGNED.ALL;
14
15 entity ClkDiv is
16     Port (  InByte : in STD_LOGIC_VECTOR(3 downto 0);       --<-- Seq_CPLD
17             RegSel : in STD_LOGIC_VECTOR(1 downto 0);       --<-- Seq_CPLD
18             RegStrb : 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          ADCClk <= '0';
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
```

Drive PWM control

IO Board


Proximity / Light Sensors
Ultrasonic ranger
Hall Sensors
Secondary
End of turn
12 V
GND
Digital Outputs

Avoid Hardware Damages

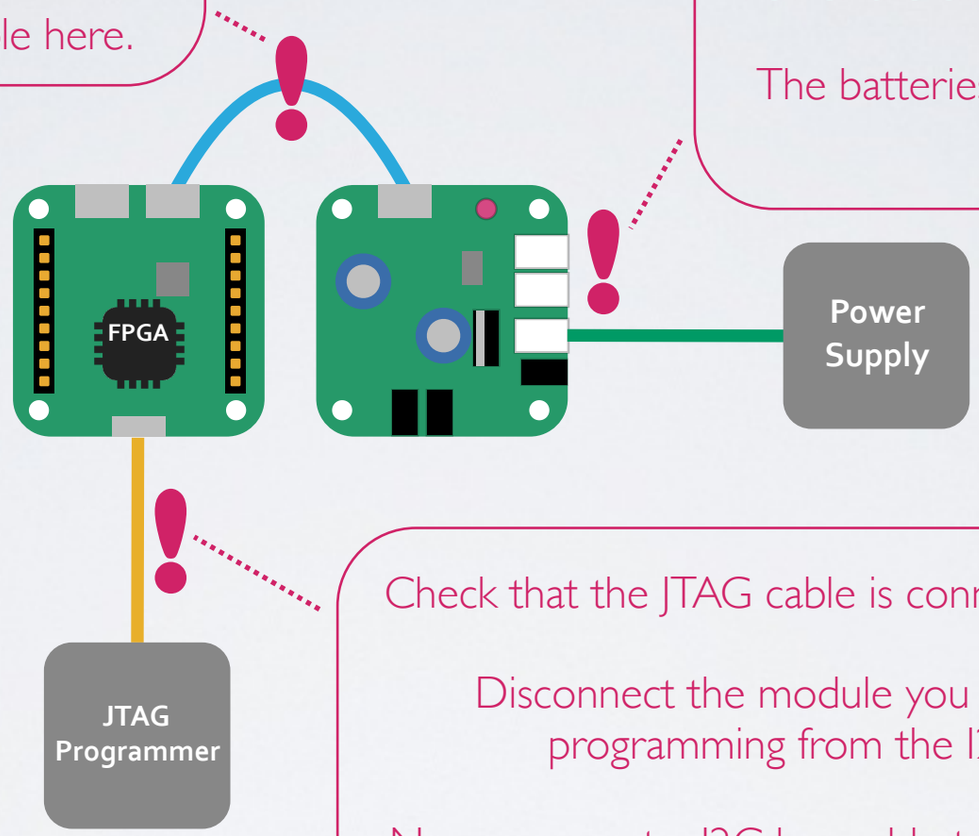
Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Double-check that the cable/connector are connected correctly.

Never connect JTAG connector cable here.

Power the circuit using a laboratory power supply.

The batteries should never be connected during development!



Check that the JTAG cable is connected correctly!

Disconnect the module you are actually programming from the I2C bus!

Never connect a I2C bus cable to this connector!

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

→ 4.0

+

1 ← **Per optional feature**

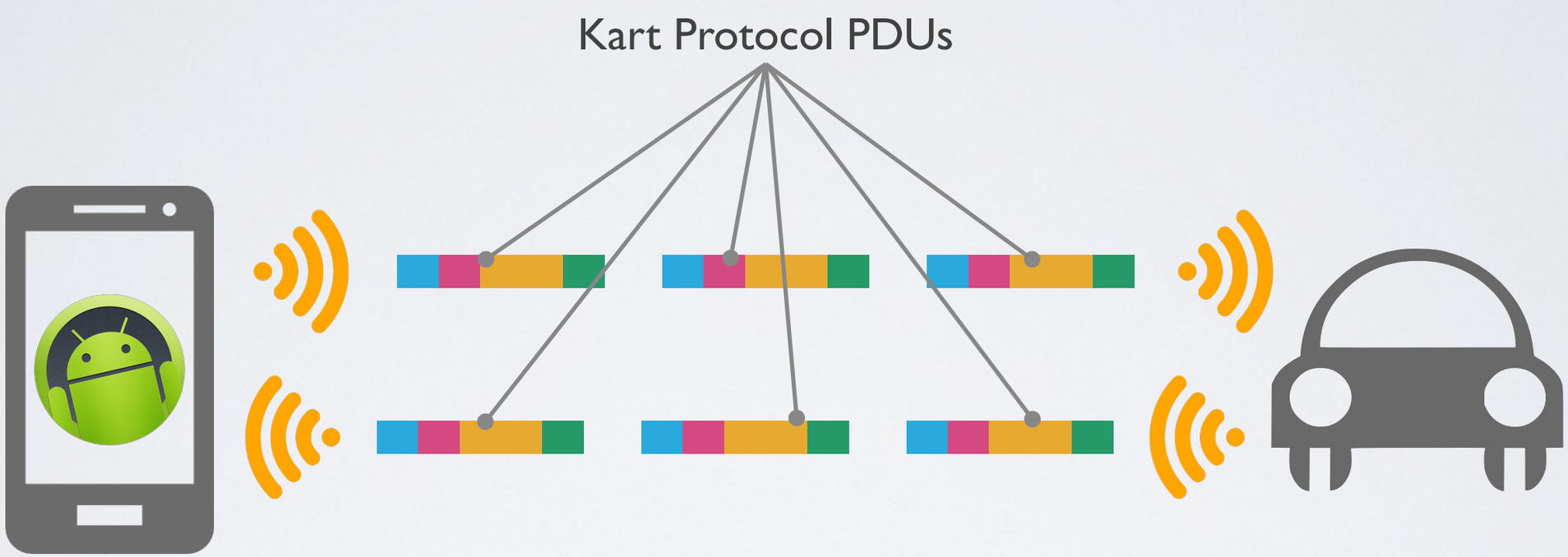
Ultrasound sensor
Emergency Stop (Proximity sensor)
Other improvements

=

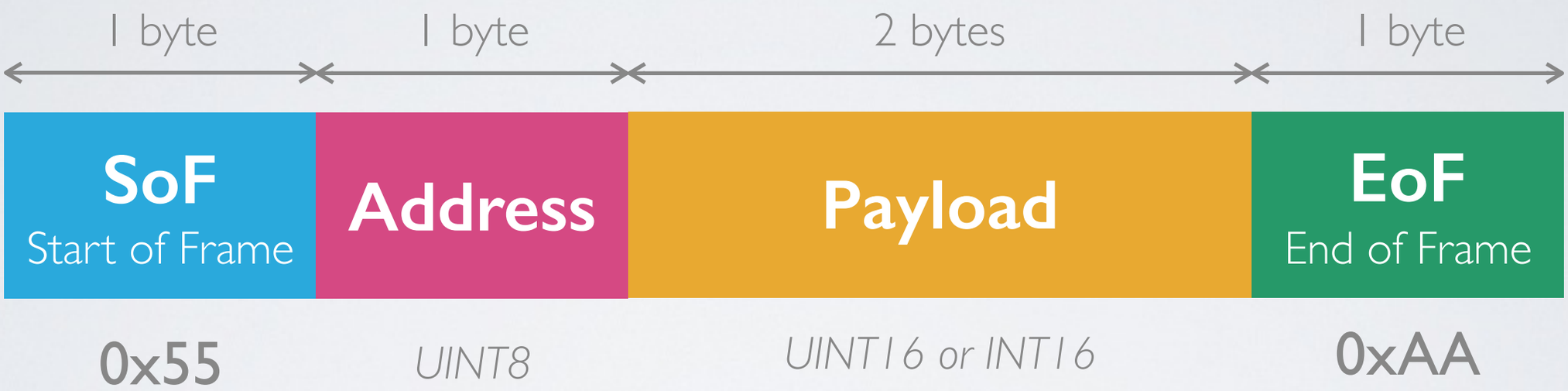
Grade

INF part

Remote Control Protocol



Remote Control Protocol PDU



Remote Control Protocol Addresses

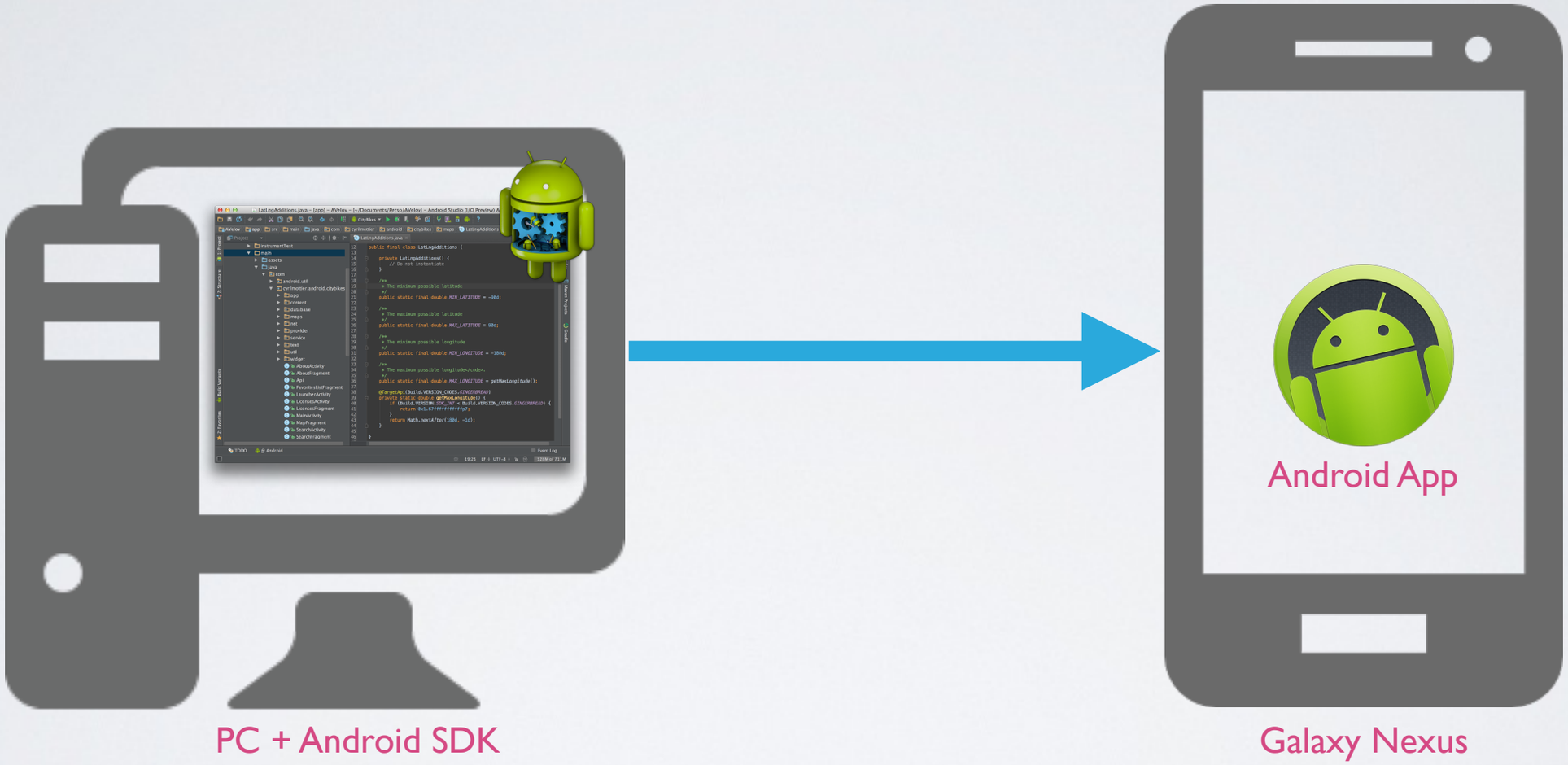
Address	Type	Description
0x00	UINT16	Drive motor PWM period
0x01	INT5	Drive motor speed [-15..15] (negative=backwards)
0x02	UINT16	Steering motor step period (speed proportional to 1/period)
0x03	UINT16	Steering position set point
0x04	UINT5	Steering end switch address
0x05	UINT5	Hardware settings (Enumeration mask)
0x06	UINT4	LED control
0x07	UINT2	Sequence control register
0x08	UINT16	Sequence operations FIFO register
0x15	UINT16	Update interval (from kart to phone) in ms

Remote Control Protocol Addresses

Address	Type	Description
0x00	UINT16	Hall sensor 1 speed count
0x01	UINT16	Hall sensor 2 speed count
0x02	UINT1	Steering angle reached (1=reached, 0=busy)
0x03	UINT16	Actual steering position
0x04	UINT1	Steering end contact state (0=contact closed)
0x05	UINT16	ADC value of battery voltage level
0x06	UINT16	Distance (Ultrasonic sensor)
0x07	UINT16	Sequence status register
0x08	UINT16	Proximity 1 (IR sensor)
0x09	UINT16	Proximity 2 (IR sensor)
0x0A	UINT16	Proximity 3 (IR sensor)
0x0B	UINT16	Proximity 4 (IR sensor)
0x0C	UINT16	Ambient Light 1 (IR sensor)
0x0D	UINT16	Ambient Light 2 (IR sensor)
0x0E	UINT16	Ambient Light 3 (IR sensor)
0c0F	UINT16	Ambient Light 4 (IR sensor)

Remote Control Android App

Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

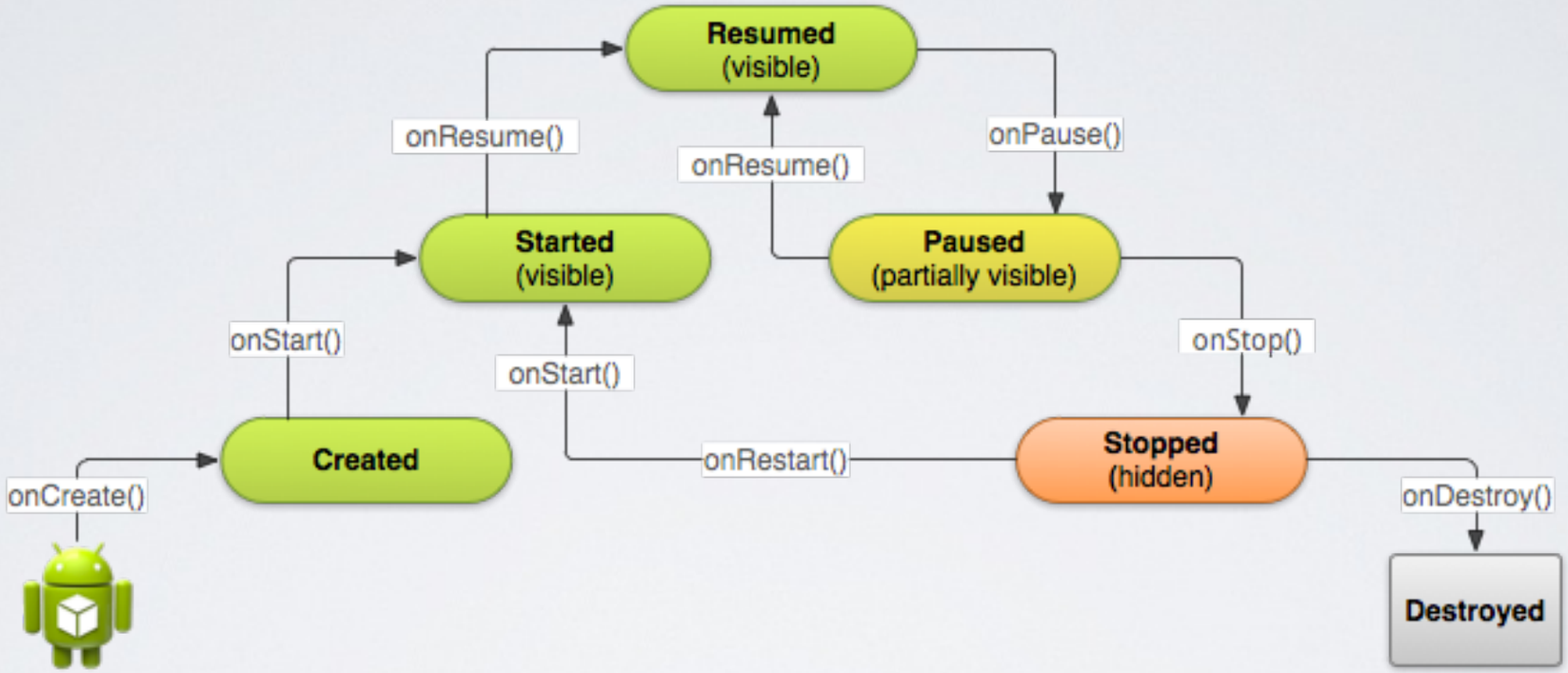


Android

- 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.

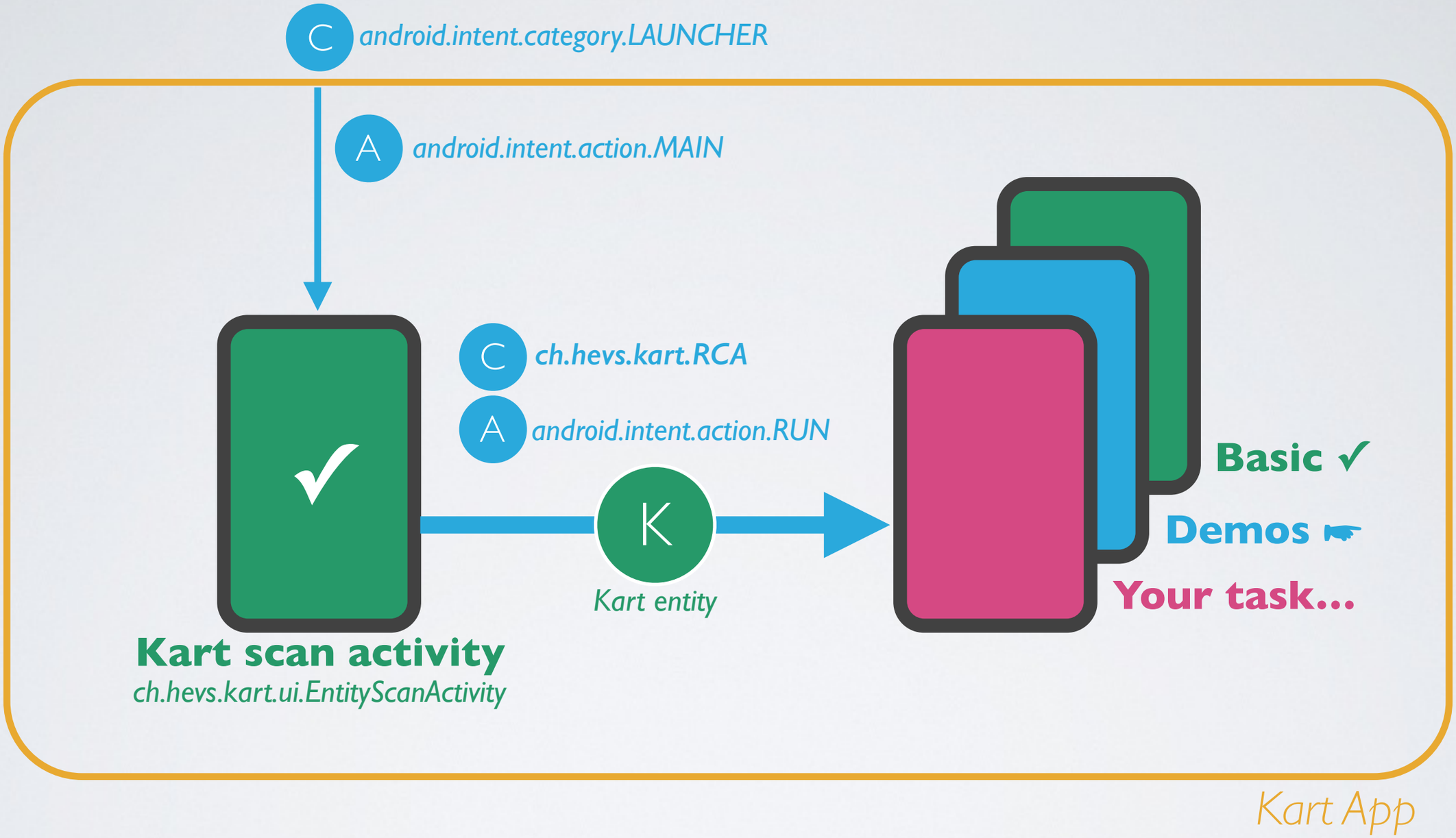
Android

- An Android application has a lifecycle.

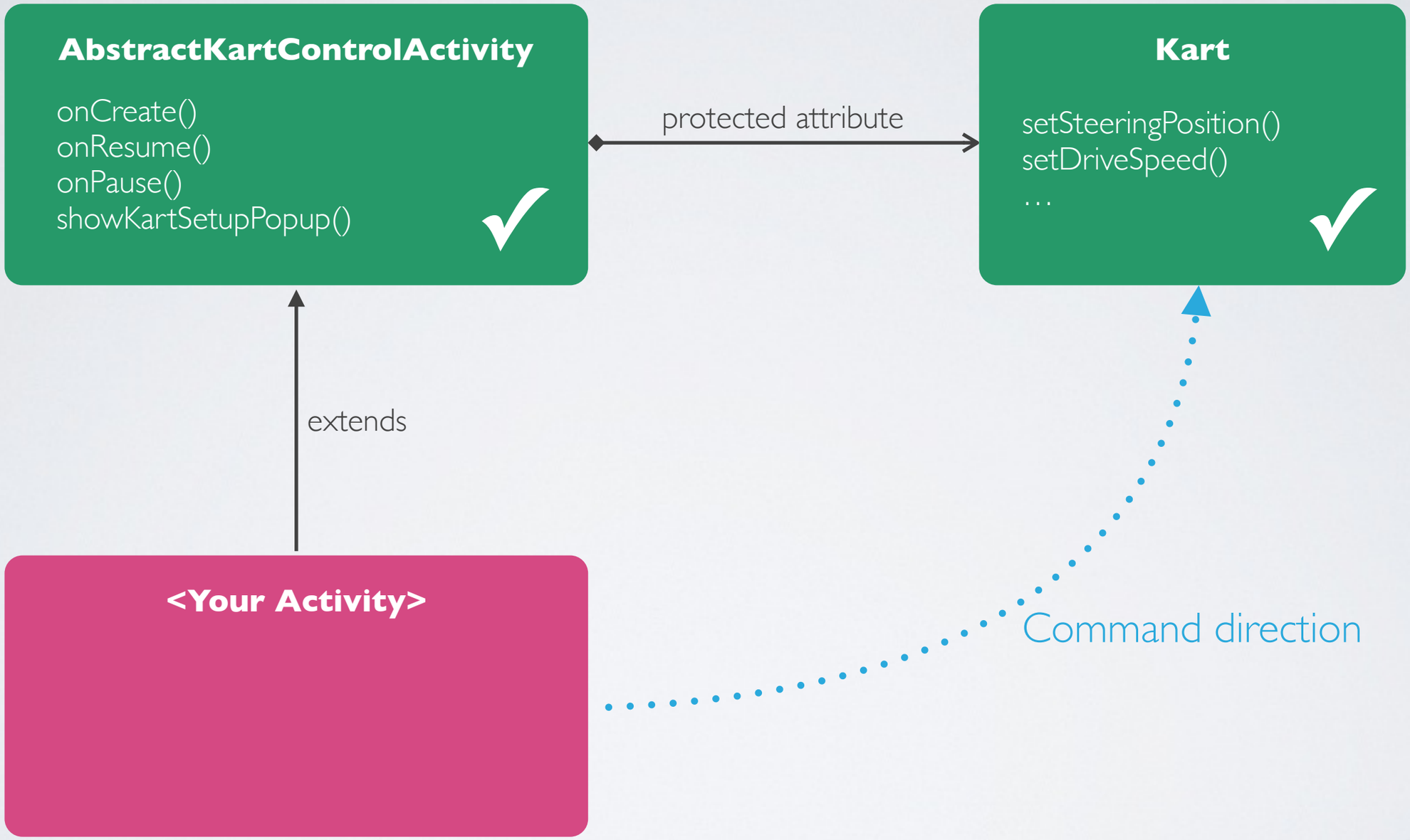


- UI layouts can be designed using an editor integrated into Android Studio.
  - Layouts are serialized to XML files.
  - Those Layouts can be loaded from code.

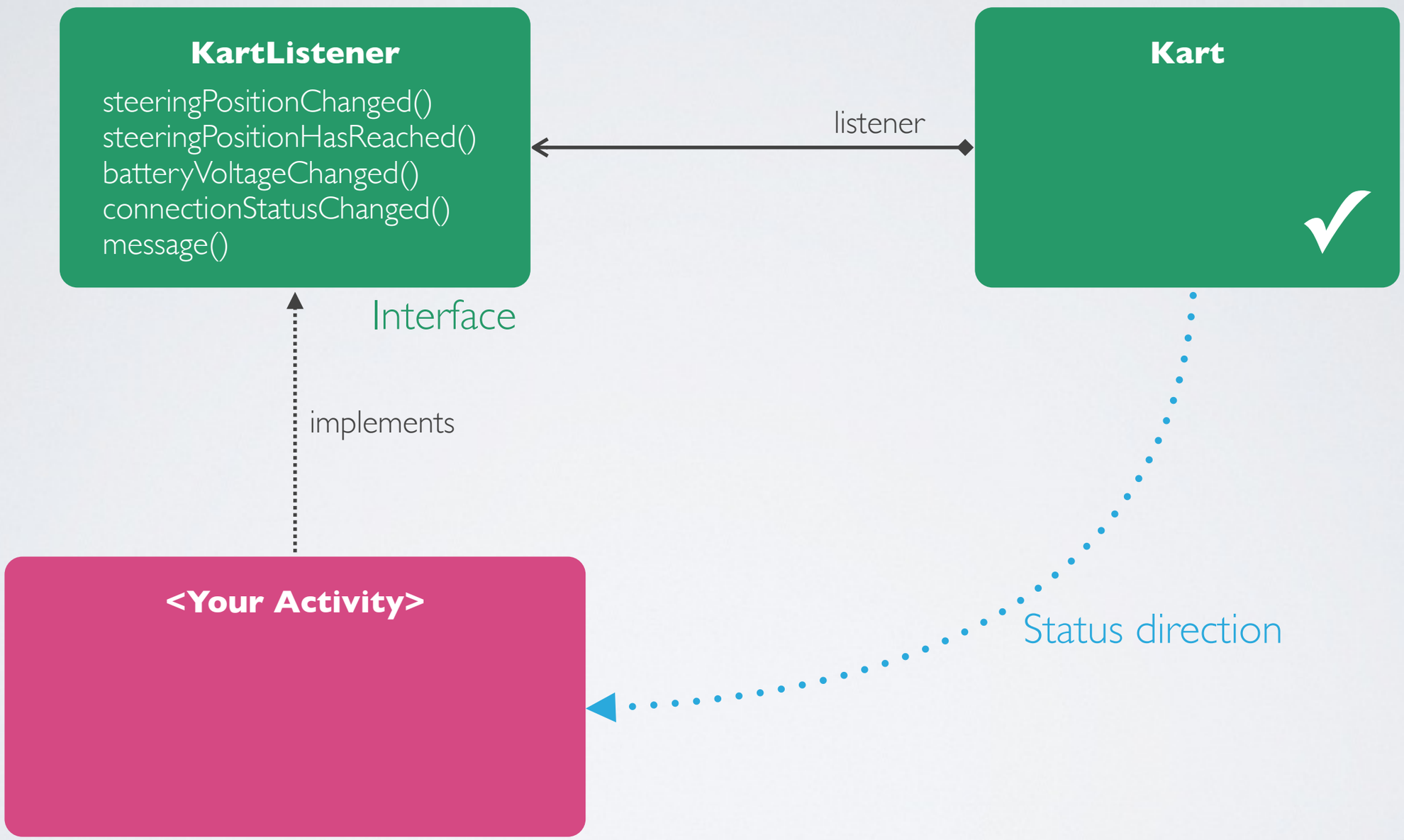
Remote Control Android App



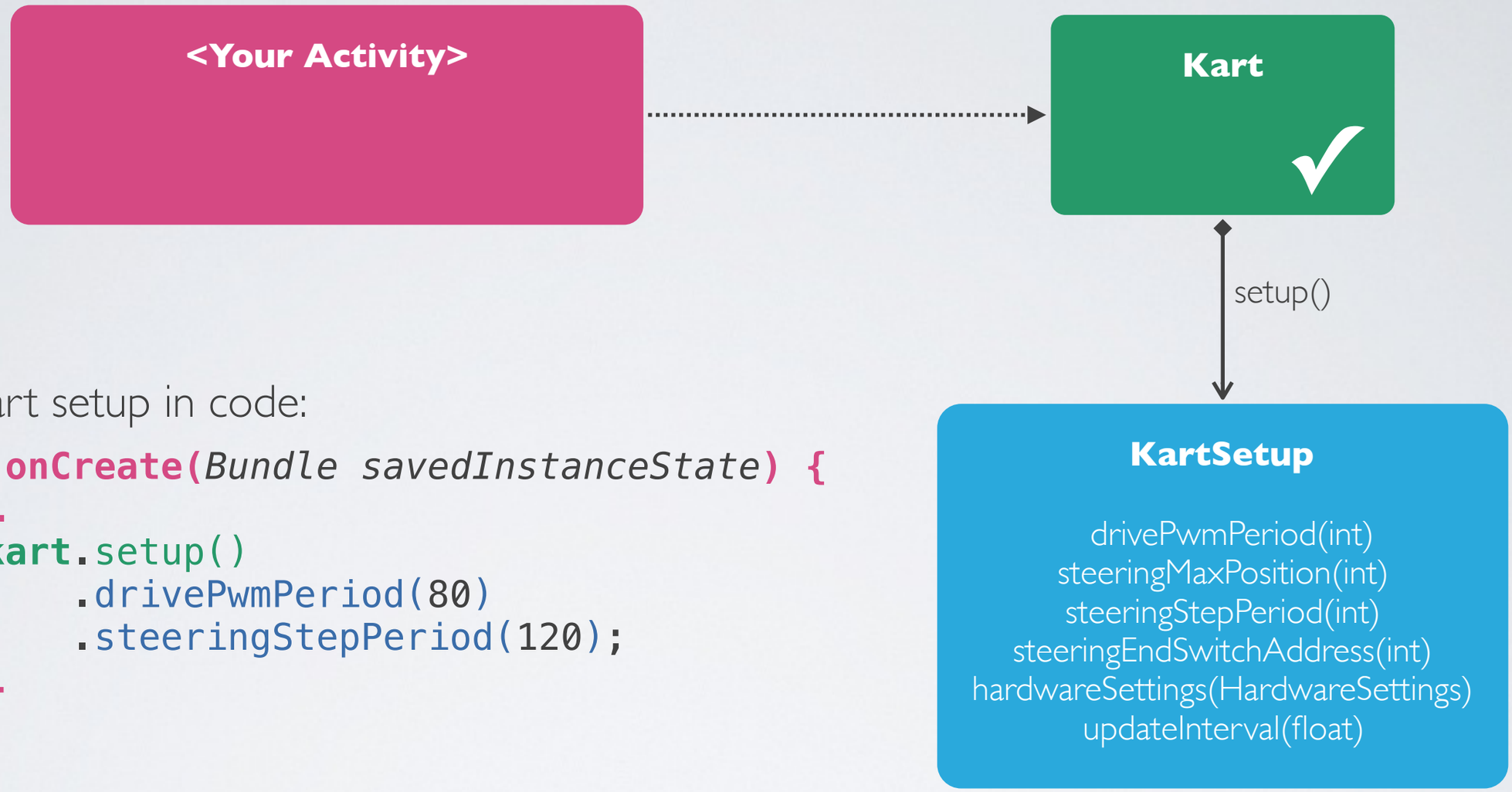
Remote Control Android App



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

→ 4.0

+

0.5 ← **Per optional feature**

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

=

Grade

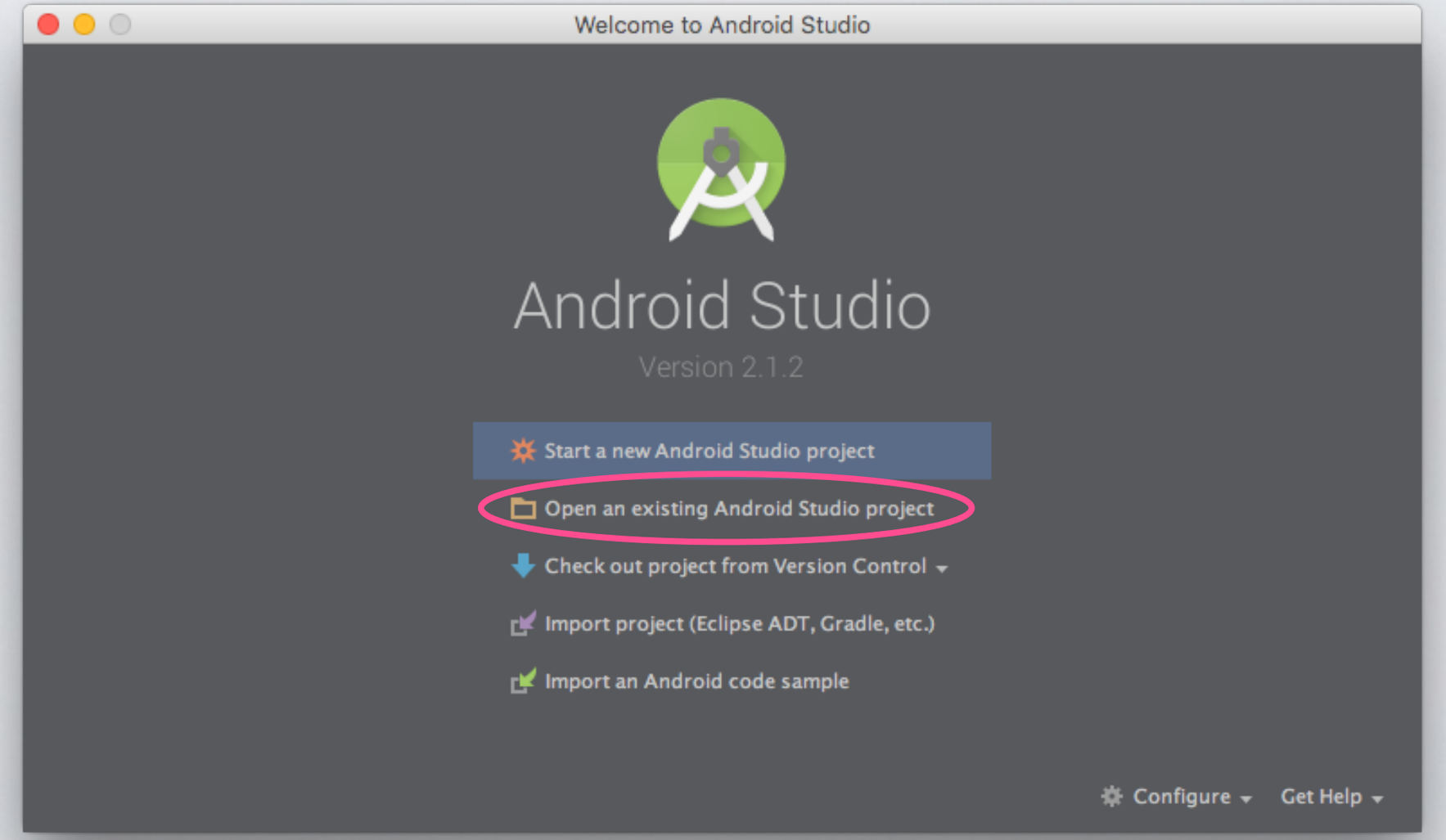
Android Studio

Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Download the Kart project template from the wiki

Extract the archive to the local disk

Open the extracted folder in Android Studio



Galaxy Nexus

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

Install & Run Demo

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
```

or much simpler:

Point your browser to: bit.ly/kart-app, download the app select it and follow the instructions to install the app.

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!

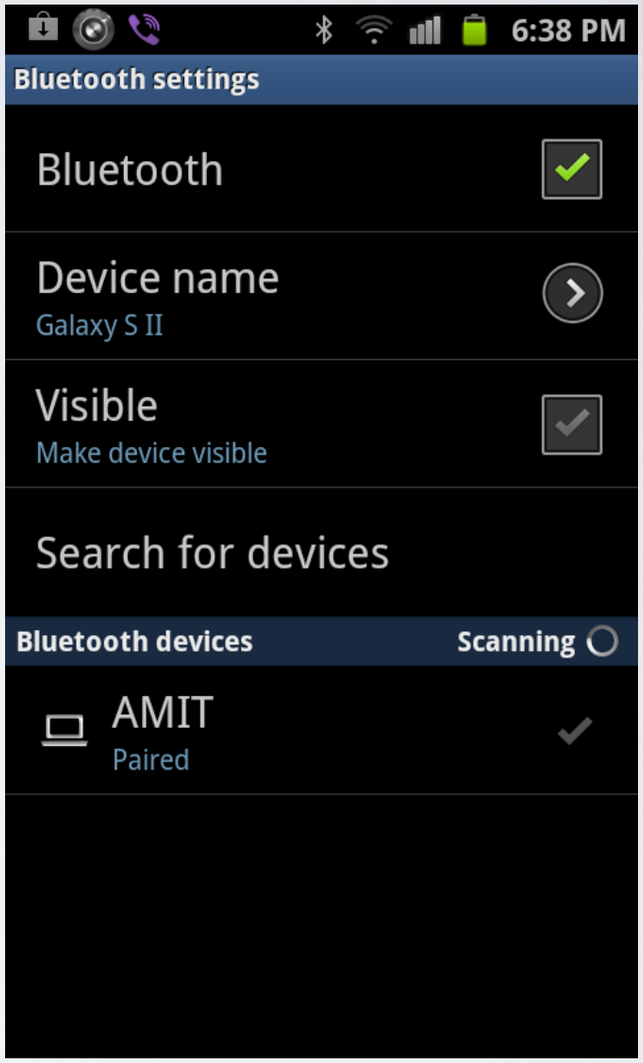
Tipp #2: Pair with your kart

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

Open "Settings"

Go to "Bluetooth"

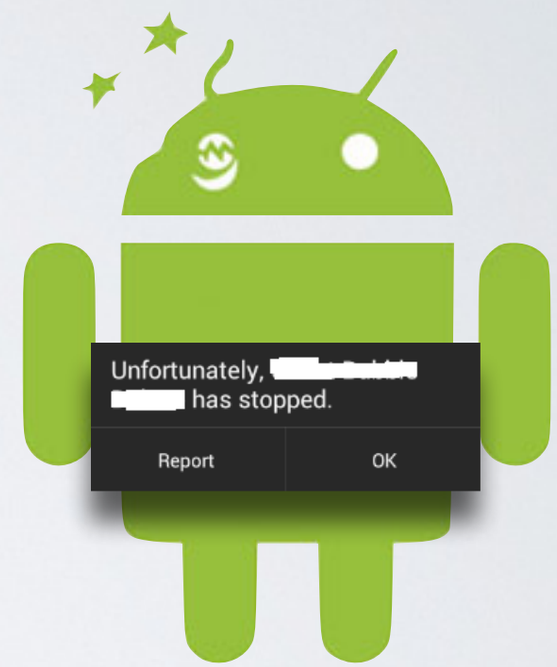
Select your kart in "Available Devices"



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

Tipp #7: Enumerations

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

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

DayofWeek.java

```java
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

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

Program output

Tipp #7: Enumerations


```java
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

```java
System.out.println("KartControlRegister has " + KartControlRegister.COUNT + " registers. ");

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

Main.java