TIZEN™

Tizen.IoT

Things에 연결된 센서로부터 센서 값 읽어오기

Tizen Platform Lab.

Tizen Talks

TIZEN

TIZEN™

Tizen.IoTivity.Sensing

Tizen Talks

TIZEN

1

Sensing device and Cloud

2

Connecting a Infrared motion sensor to RPi3

How to connect pins between Infrared motion sensor and RPi3

3

Code structure

- RCC Pattern
- Lifecycles

4

Implementing the code

How to read/write values from the sensor

1

Sensing device and Cloud

2

Connecting a Infrared motion sensor to RPi3

How to connect pins between Infrared motion sensor and RPi3

3

Code structure

- RCC Pattern
- Lifecycles

4

Implementing the code

How to read/write values from the sensor

Sensing device and Cloud

Sensing device

Cloud Server

Details of sensing device

1. Data Gathering

Be able to recognize circumstance

Using
Infrared
Motion Sensor

2. Programmable

Be able to control data

Using
Raspberry Pi3

3. Networkable

Be able to communicate with Cloud server

Using
Wi-Fi chipset

1

Sensing device and Cloud

2

Connecting a Infrared motion sensor to RPi3

How to connect pins between Infrared motion sensor and RPi3

3

Code structure

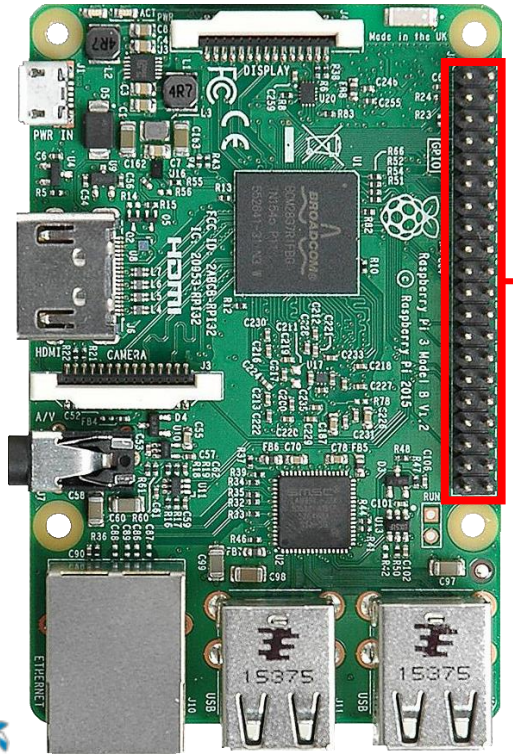
- RCC Pattern
- Lifecycles

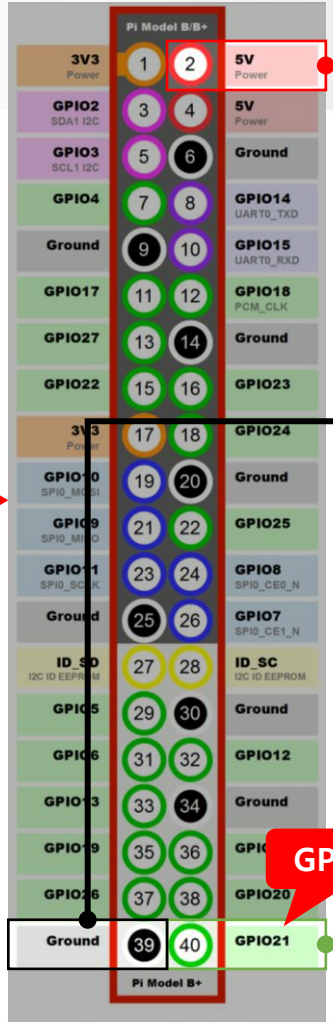
4

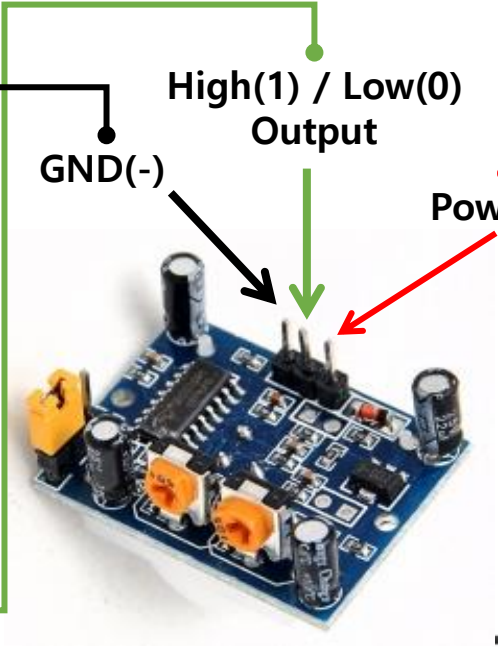
Implementing the code

How to read/write values from the sensor

Connecting sensor to RPi3







Source : www.raspberrypi-spy.co.uk

1

Sensing device and Cloud

2

Connecting a Infrared motion sensor to RPi3

How to connect pins between Infrared motion sensor and RPi3

3

Code structure

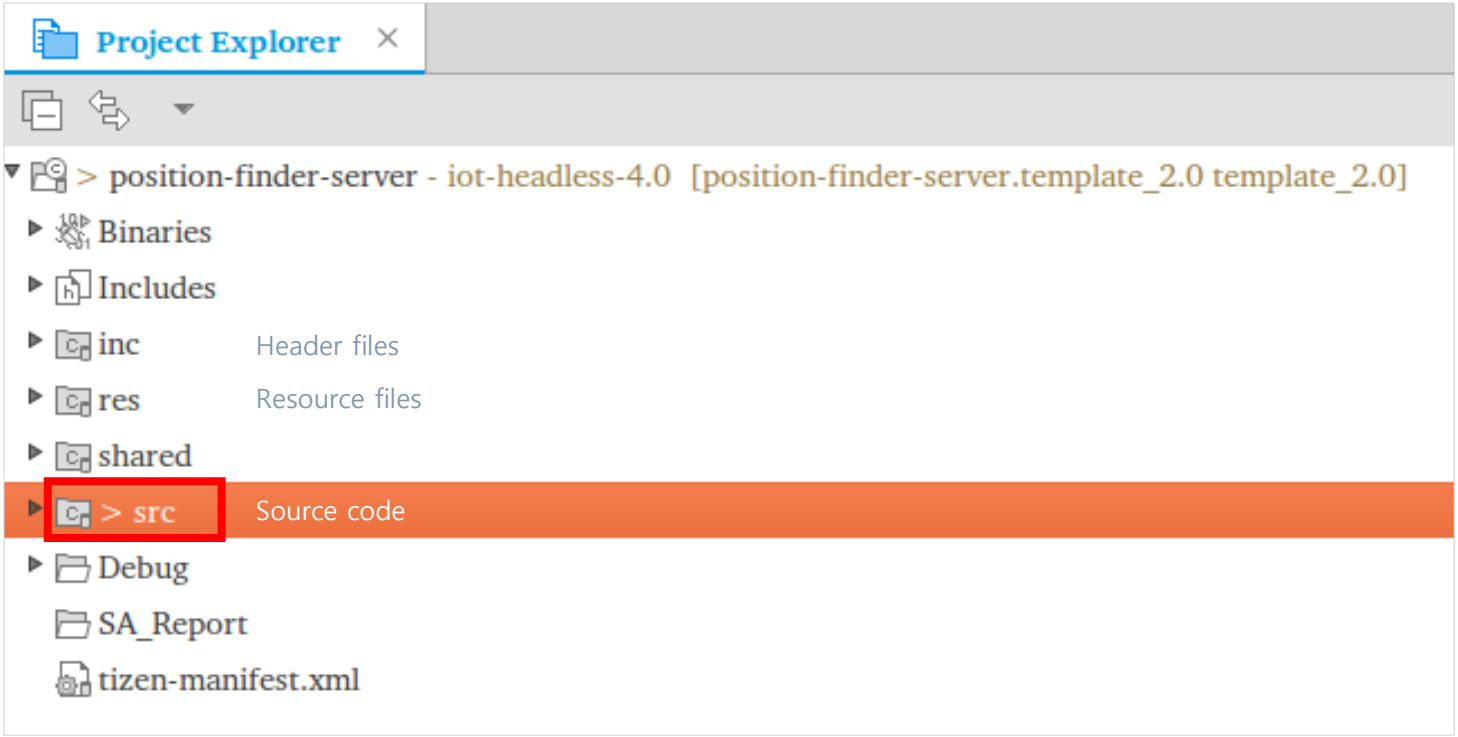
- RCC Pattern
- LifeCycles

4

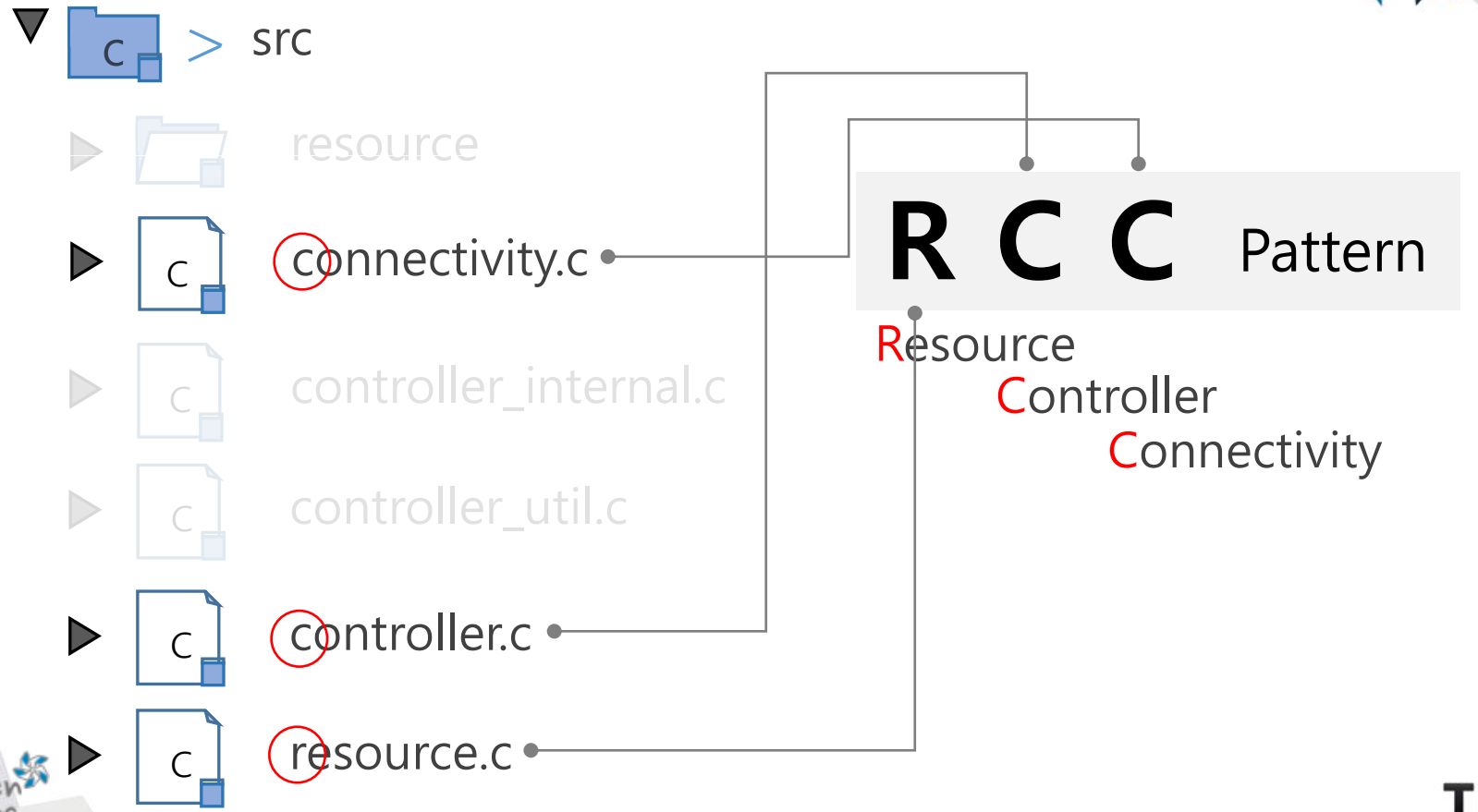
Implementing the code

How to read/write values from the sensor

RCC Pattern


Project Explorer
> position-finder-server - iot-headless-4.0 [position-finder-server.template_2.0 template_2.0]
Binaries
Includes
inc
Header files
res
Resource files
shared
> src
Source code
Debug
SA_Report
tizen-manifest.xml

RCC Pattern

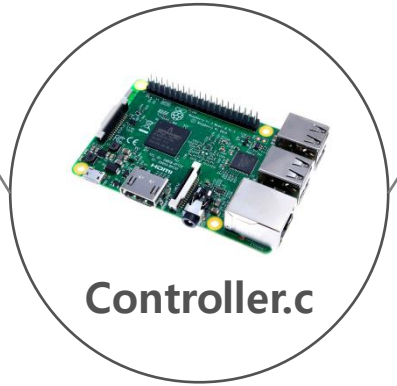

src
resource
connectivity.c
controller_internal.c
controller_util.c
controller.c
resource.c
R C C Pattern
Resource
Controller
Connectivity

RCC Pattern

Information of the environment can be received/sent as data using sensors, LED etc.

Connects to a device connected to local network or cloud server

Tizen IoTivity Application start from here !



Controls resources and connectivity

Tizen Service Application LifeCycles

Position-finder-server/src/controller.c

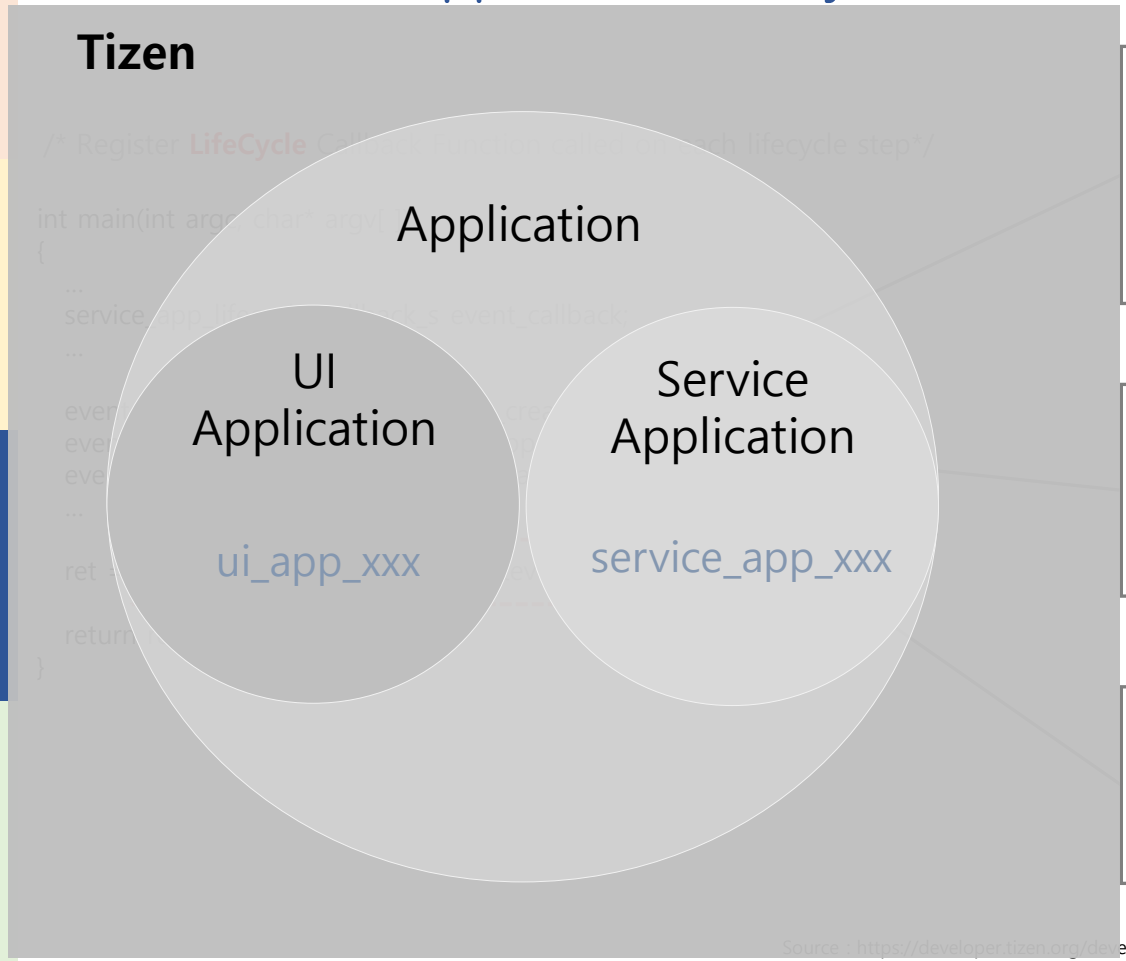
```
/* Register LifeCycle Callback Function called on each lifecycle step*/

int main(int argc, char* argv[ ])
{
    ...
    service_app_lifecycle_callback_s event_callback;
    ...

    event_callback.create = service_app_create;
    event_callback.terminate = service_app_terminate;
    event_callback.app_control = service_app_control;
    ...

    ret = service_app_main(argc, argv, &event_callback, ad);

    return ret;
}
```

```
static bool service_app_create(void *data)
{
    ...

    return true;
}
```

```
static void service_app_terminate(void *data)
{
    ...
}
```

```
static void service_app_control(void *data)
{
    ...
}
```

Tizen Service Application LifeCycles



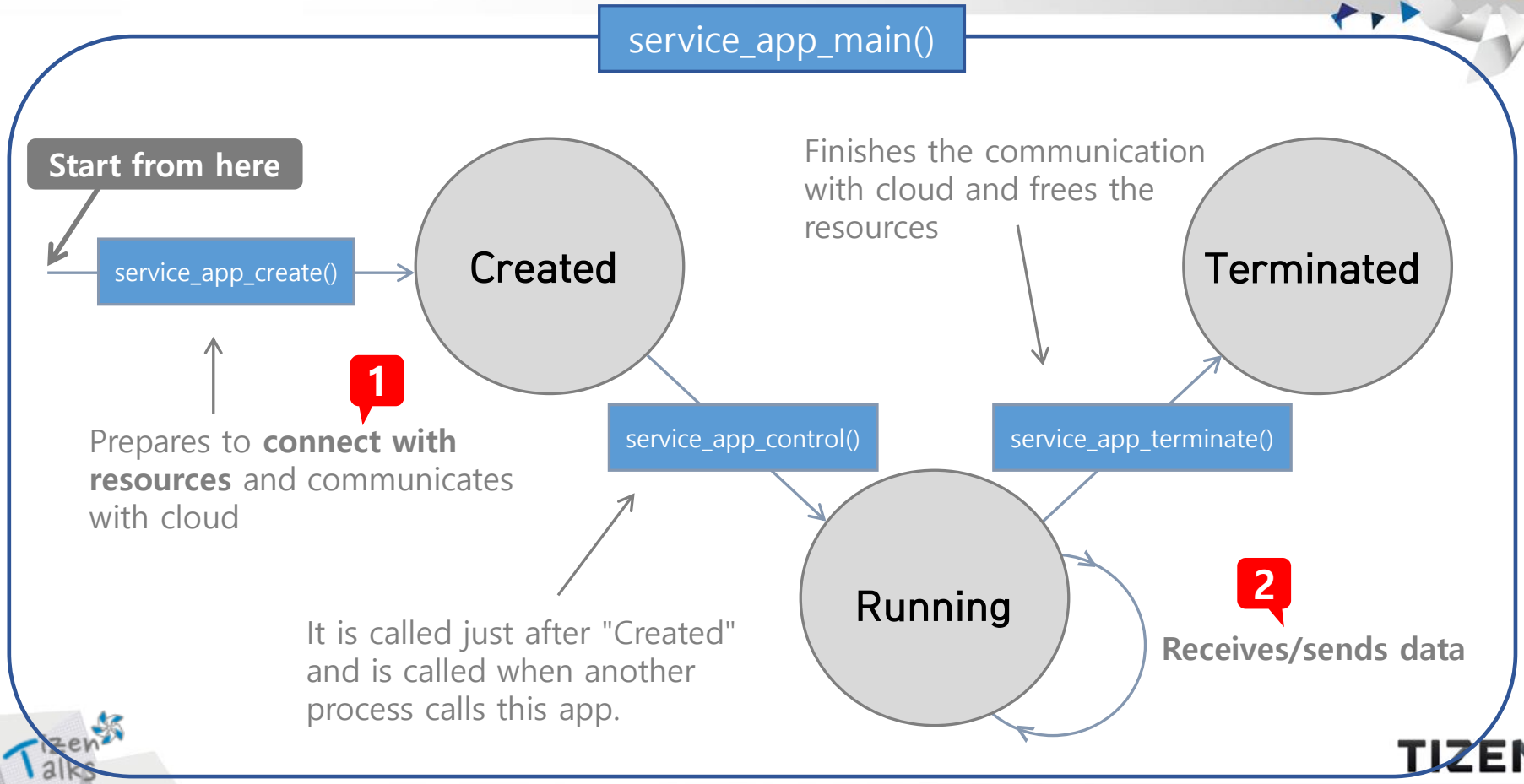

```
static bool service_app_create(void *data)
{
    ...

    return true;
}
```

```
static void service_app_terminate(void *data)
{
    ...
}
```

```
static void service_app_control(void *data)
{
    ...
}
```

Source : https://developer.tizen.org/development/guides/web-application/application-management/applications/service-application

Tizen Service Application LifeCycles



Source : https://developer.tizen.org/development/guides/web-application/application-management/applications/service-application

1

Sensing device and Cloud

2

Connecting a Infrared motion sensor to RPi3

How to connect pins between Infrared motion sensor and RPi3

3

Code structure

- RCC Pattern
- Lifecycles

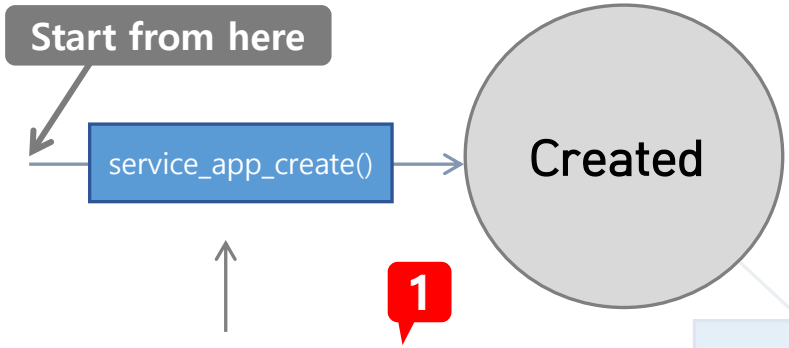
4

How to implement resources

How to read/write values from the sensor

service_app_create()

service_app_main()



Prepares to **connect with resources** and communicates with cloud

```
static bool service_app_create(void *data)
{
    app_data *ad = data;
    int ret = -1;

    /**
     * DO NOT EDIT: please don't edit the function below.
     * Access only when modifying internal functions.
     */
    controller_init_internal_functions();

    ...

    /**
     * Creates a timer to call the given function in the given period of time.
     * In the control_sensors_cb(), each sensor reads the measured value
     * or writes a specific value to the sensor.
     */
#if 0
    ad->getter_timer = ecore_timer_add(Duration , _control_sensors_cb, ad);
    if (!ad->getter_timer) {
        _E("Failed to add getter timer");
        return false;
    }
#endif

    return true;
}
```

service_app_create()

service_app_main()

Start from here

service_app_create()

Created

Prepares to con
resources and c
with cloud

How can we connect with resources?

```
static bool service_app_create(void *data)
{
    app_data *ad = data;
    int ret = -1;

    /**
     * DO NOT EDIT: please don't edit the function below.
     * Access only when modifying internal functions.
     */
    controller_init_internal_functions();

    ...

    /**
                                                    tion in the given period of time.
                                                    sor reads the measured value
                                                    sor.

                                                    _control_sensors_cb, ad);

#endif

    return true;
}
```

Reference – Ultrasonic distance sensor(HC-SR04)

Let's search for the Ultrasonic sensor(HC-SR04) datasheet…

4 wires are needed
GPIO or I2C available

Input wire / Output wire are different
Set proper type(In/Out) to each wire

Just detect high pulse
There's no larger data more than binary

• Module Pin Definitions

Pin Symbol	Pin Function Description
VCC	5V power supply
Trig	Trigger input pin of Sensor
Echo	Echo output pin of Sensor
GND	Power ground

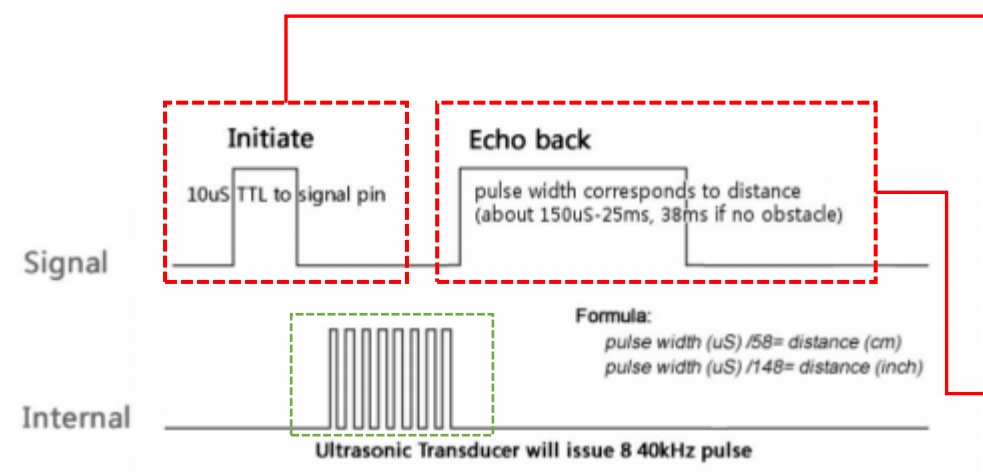
• Module Operating Principle

Set low the Trig and Echo port when the module initializes, firstly, transmit at least 10us high level pulse to the Trig pin (module automatically sends eight 40K square wave), and then wait to capture the rising edge output by echo port, at the same time, open the timer to start timing. Next, once again capture the falling edge output by echo port, at the same time, read the time of the counter, which is the ultrasonic running time in the air.

Source : https://elecfreaks.com/estore/download/EF03085-HC-SR04_Ultrasonic_Module_User_Guide.pdf, http://www.micropik.com/PDF/HCSR04.pdf

Reference – Ultrasonic distance sensor(HC-SR04)

- **Timing Diagram**



To start the sensor

1. Ensure that the Trigger pin is set low for a second
2. Set out trigger pin high for 10us to start the ranging program (8 ultrasound bursts at 40kHz)

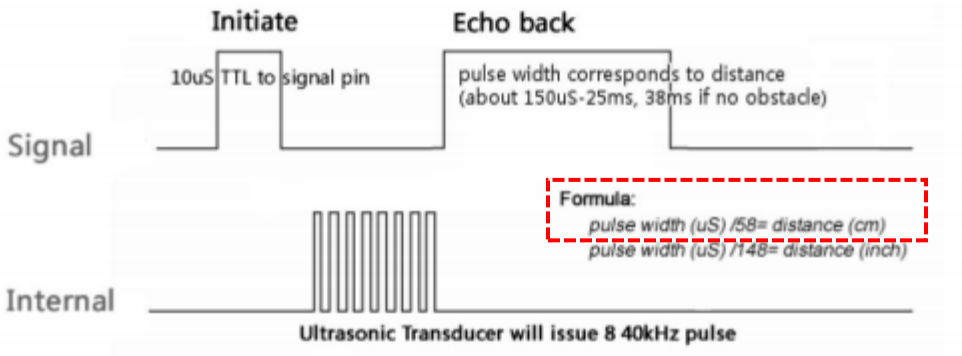
Receive the return signal

1. As soon as Trigger pin make burst, the sensor set Echo pin to high
2. If Echo pin takes the pulse that come back, sensor set Echo pin to low

Source : https://elecfreaks.com/estore/download/EF03085-HC-SR04_Ultrasonic_Module_User_Guide.pdf, http://www.micropik.com/PDF/HCSR04.pdf

Reference – Ultrasonic distance sensor(HC-SR04)

- **Formula**



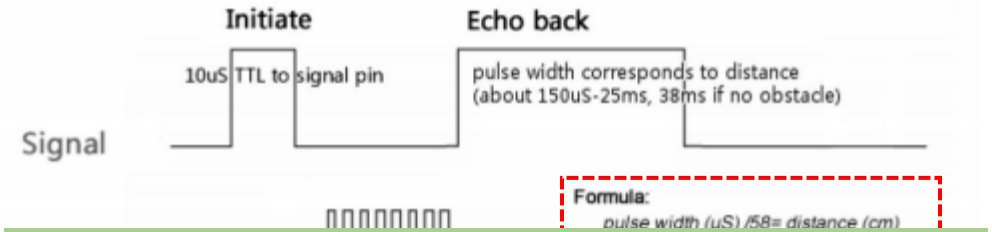
Distance = (high level time * ultrasonic spreading velocity in air) / 2

** Pulse width = 2 x distance (distance to object) / 340(m/s)*
** The sound velocity is approximately 340m/s.*

Distance(cm) = pulse width(uS) / 58

Reference – Ultrasonic distance sensor(HC-SR04)

- **Timing Diagram**



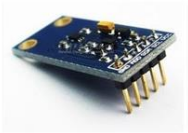
There's too much to know.
So, We have prepared !

Dist ir) / 2

** Pulse width = 2 x distance (distance to object) / 340(m/s)*
** The sound velocity is approximately 340m/s.*

Distance(cm) = pulse width(uS) / 58

Src/resource

- src
  - resource
    - resource_illuminance_sensor.c 
    - resource_infrared_motion_sensor.c 
    - resource_infrared_obstacle_avoidance_sensor.c 
    - resource_led.c 
    - resource_ultrasonic_sensor.c 
    - ...

How to read/write values from the sensor

1. Checks the function name with sensor name you want to use in **resource** directory.

2. Determines the **time** to read/write sensor value.

3. Reads/writes value from/to sensor.

1. Checks the function name with sensor name you want to use in resource directory.

- src
  - resource
    - resource_illuminance_sensor.c
    - resource_infrared_motion_sensor.c
    - resource_infrared_obstacle_avoidance_sensor.c
    - resource_led.c
    - resource_ultrasonic_sensor.c
    - ...

2. Determines the time to read/write sensor value.

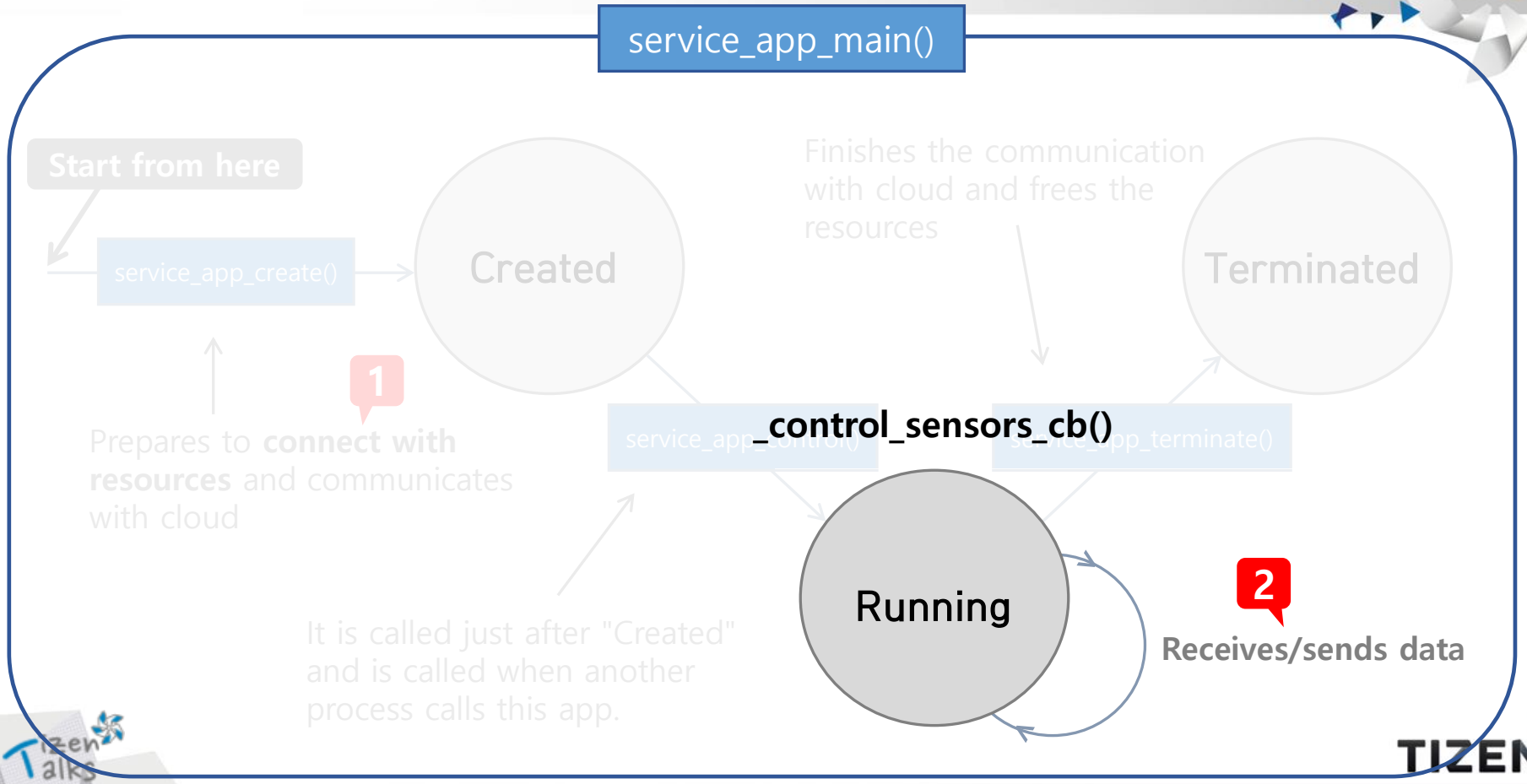
service_app_create()

```
static  bool service_app_create(void *data)
{
    ...
  #if 1
    ad->getter_timer = ecore_timer_add(Duration , _control_sensors_cb, ad);
    if (!ad->getter_timer) {
        _E("Failed to add getter timer");
        return false;
    }
  #endif

    return true;
}
```

Type: float e.g.) 2.0f

This parameter decides how often callback function will be called.

Runnig step : _control_sensors_cb



Source : https://developer.tizen.org/development/guides/web-application/application-management/applications/service-application

3. Reads/writes value from/to sensor.

_control_sensors_cb()

```
static  Eina_Bool _control_sensors_cb(void *data)
{
   ...

   ...

   return true;
}
```

resource/resource_infrared_motion_sensor.c

```
void resource_close_infrared_motion_sensor(int pin_num)
{
   ...
}

int resource_read_infrared_motion_sensor(int pin_num, int *out_value)
{
   int ret = PERIPHERAL_ERROR_NONE;

   ...
   ret = peripheral_gpio_read(resource_get_info(pin_num)->sensor_h, out_value);
   retv_if(ret != PERIPHERAL_ERROR_NONE, -1);

   return 0;
}
```

3. Reads/writes value from/to sensor.

control_sensors_cb()

```
static  Eina_Bool control_sensors_cb(void *data)
{
   ...

   ...

   return true;
}
```

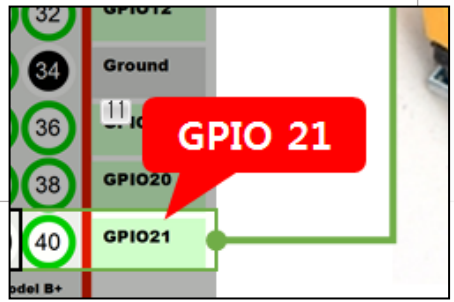
resource/resource_infrared_motion_sensor.c

```
void resource_close_infrared_motion_sensor(int pin_num)
{
   ...
}

int resource_read_infrared_motion_sensor(int pin_num, int *out_value)
{
   int ret = PERIPHERAL_ERROR_NONE;

   ...
   ret = peripheral_gpio_read(resource_get_info(pin_num)->sensor_h, out_value);
   retv_if(ret != PERIPHERAL_ERROR_NONE, -1);

   return 0;
}
```

3. Reads/writes value from/to sensor.

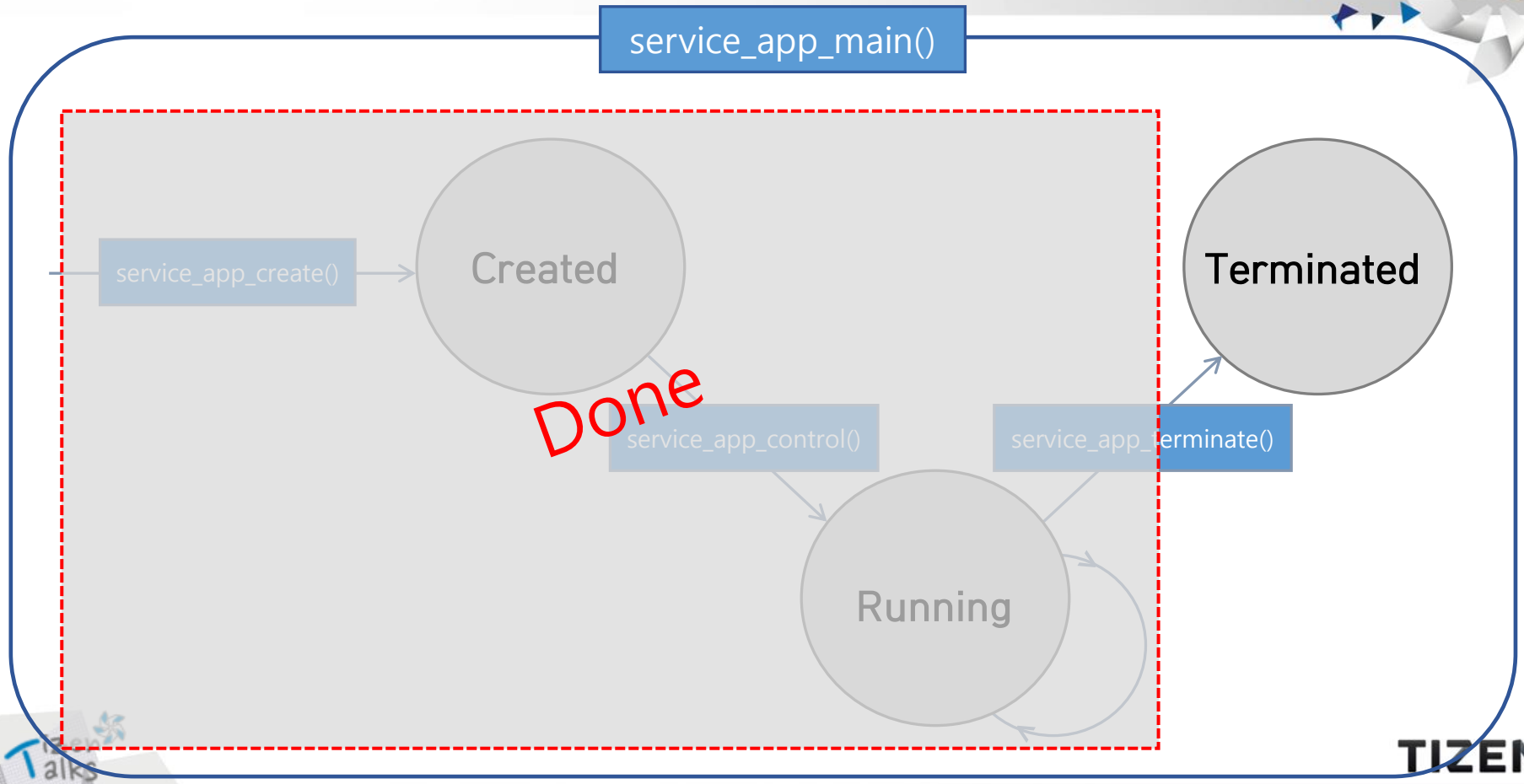
_control_sensors_cb()

```
static  Eina_Bool _control_sensors_cb(void *data)
{
   ...
  #if 1
   ret = resource_read_infrared_motion_sensor(PIN_NUMBER, &value);
   if (ret != 0) _E("Cannot read sensor value");

   _D("Detected value : %d", value);

   #endif

   ...

   return true;
}
```



resource/resource_infrared_motion_sensor.c

```
void resource_close_infrared_motion_sensor(int pin_num)
{
   ...
}

int resource_read_infrared_motion_sensor(int pin_num, int *out_value)
{
   int ret = PERIPHERAL_ERROR_NONE;

   ...
   ret = peripheral_gpio_read(resource_get_info(pin_num)->sensor_h, out_value);
   retv_if(ret != PERIPHERAL_ERROR_NONE, -1);

   return 0;
```


service_app_main()
service_app_create()
Created
Done
service_app_control()
Running
service_app_terminate()
Terminated

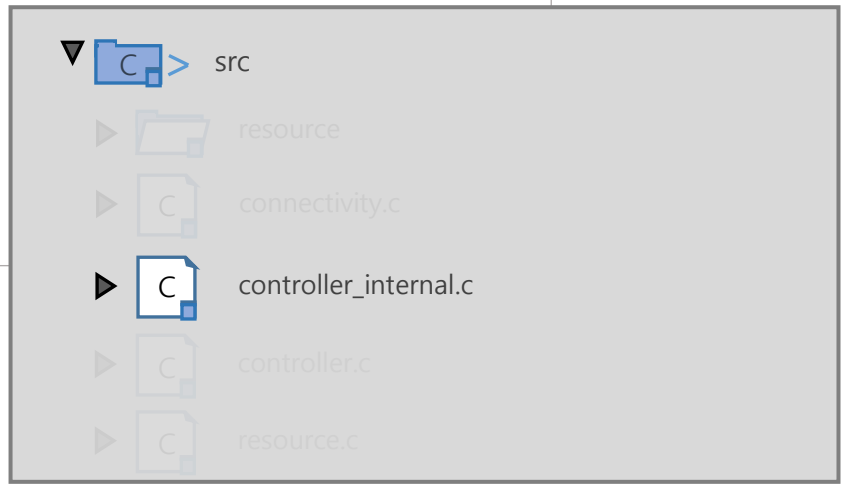
*How to close the resources?

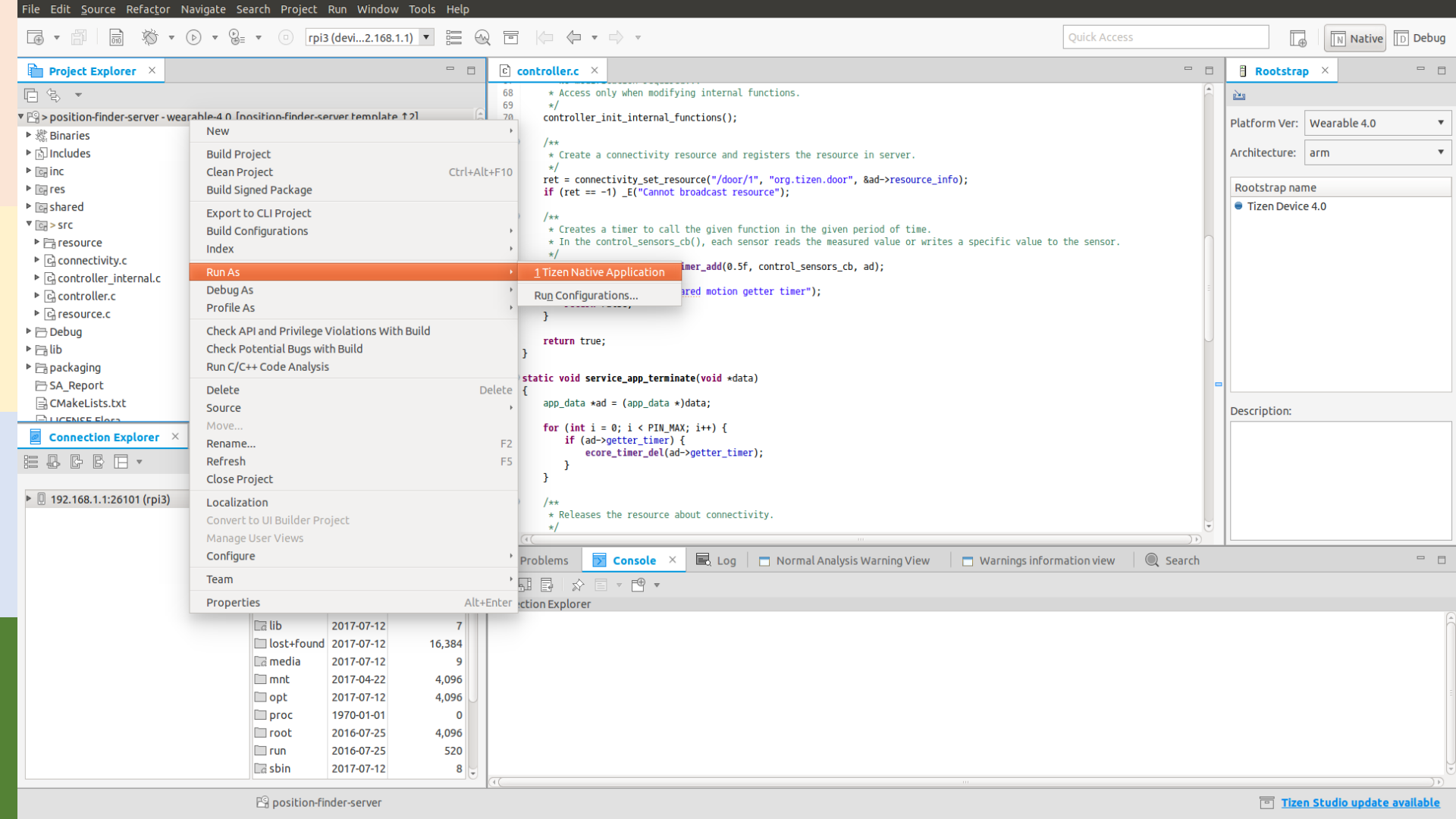
service_app_terminate()

```
static  void service_app_terminate(void *data)
{
    ...
    controller_fini_internal_function();
    ...
    ...

    return true;
}
```

All connections with resources are closed in this function

You do not have to do anything.




File Edit Source Refactor Navigate Search Project Run Window Tools Help
Quick Access
Native
Debug
Project Explorer
> position-finder-server - wearable-4.0 [position-finder-server template 1.2]
Binaries
Includes
inc
res
shared
> src
resource
connectivity.c
controller_internal.c
controller.c
resource.c
Debug
lib
packaging
SA_Report
CMakeLists.txt
New
Build Project
Clean Project Ctrl+Alt+F10
Build Signed Package
Export to CLI Project
Build Configurations
Index
Run As
Debug As
Profile As
Check API and Privilege Violations With Build
Check Potential Bugs with Build
Run C/C++ Code Analysis
Delete Delete
Source
Move...
Rename... F2
Refresh F5
Close Project
Localization
Convert to UI Builder Project
Manage User Views
Configure
Team
Properties Alt+Enter
1 Tizen Native Application
Run Configurations...
controller.c
* Access only when modifying internal functions.
controller_init_internal_functions();
* Create a connectivity resource and registers the resource in server.
ret = connectivity_set_resource("/door/1", "org.tizen.door", &ad->resource_info);
if (ret == -1) _E("Cannot broadcast resource");
* Creates a timer to call the given function in the given period of time.
* In the control_sensors_cb(), each sensor reads the measured value or writes a specific value to the sensor.
imer_add(0.5f, control_sensors_cb, ad);
ared motion getter timer");
return true;
static void service_app_terminate(void *data)
app_data *ad = (app_data *)data;
for (int i = 0; i < PIN_MAX; i++) {
if (ad->getter_timer) {
ecore_timer_del(ad->getter_timer);
* Releases the resource about connectivity.
Rootstrap
Platform Ver: Wearable 4.0
Architecture: arm
Rootstrap name
Tizen Device 4.0
Description:
Connection Explorer
192.168.1.1:26101 (rpi3)
lib 2017-07-12 7
lost+found 2017-07-12 16,384
media 2017-07-12 9
mnt 2017-04-22 4,096
opt 2017-07-12 4,096
proc 1970-01-01 0
root 2016-07-25 4,096
run 2016-07-25 520
sbin 2017-07-12 8
Problems
Console
Log
Normal Analysis Warning View
Warnings information view
Search
position-finder-server
Tizen Studio update available

TIZEN™

Thank you

Tizen Talks

TIZEN