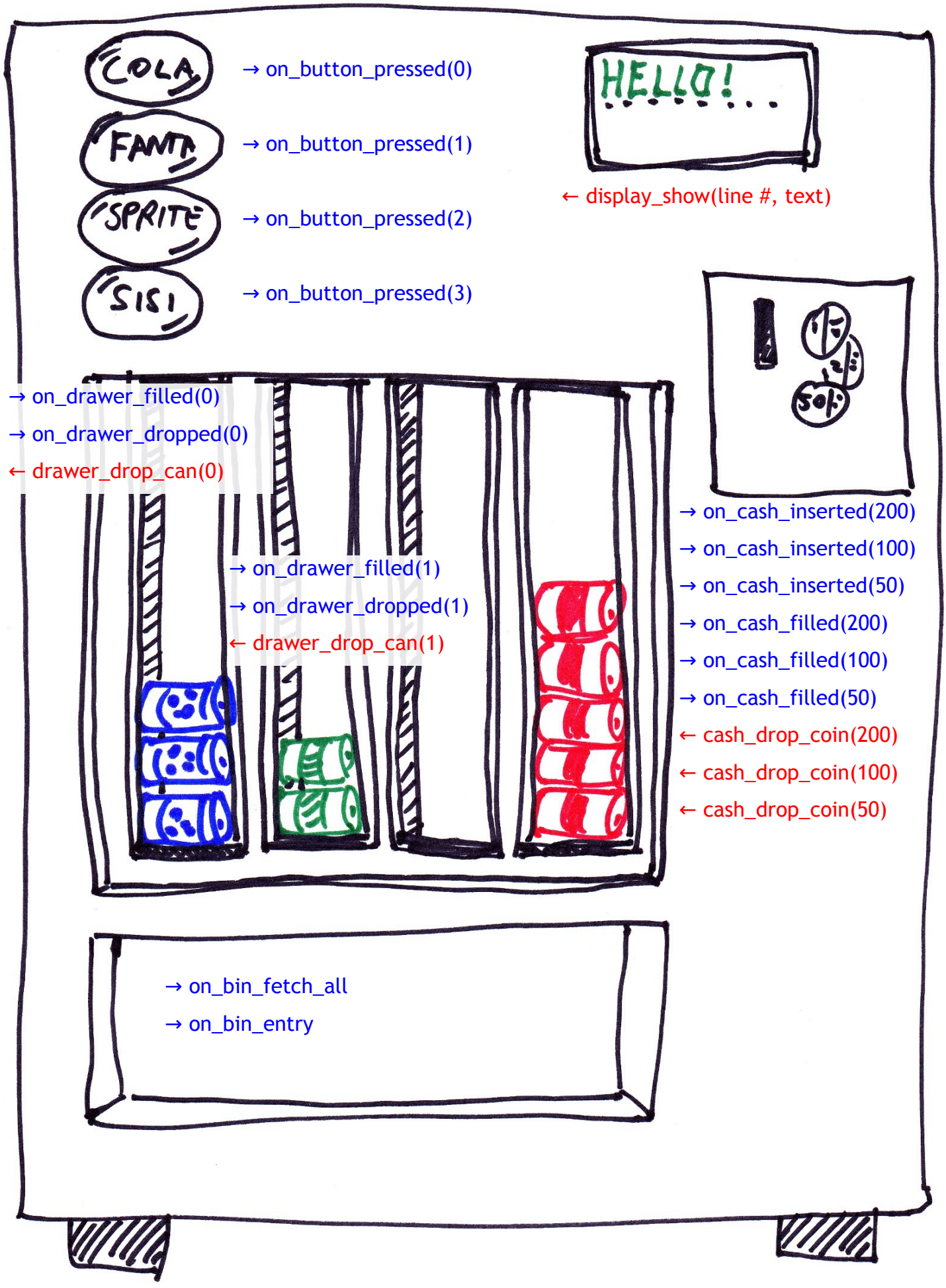
QWAN

Quality Without A Name

Handout

Embedded Vending Machine

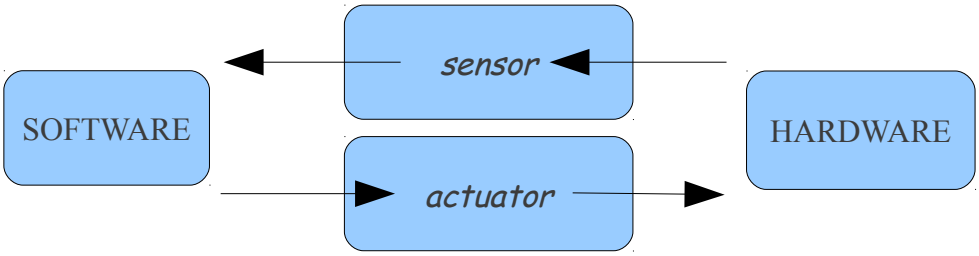
Agile Engineering Course


COLA
FANTA
SPRITE
SISI
→ on_button_pressed(0)
→ on_button_pressed(1)
→ on_button_pressed(2)
→ on_button_pressed(3)
HELLO!
← display_show(line #, text)
→ on_drawer_filled(0)
→ on_drawer_dropped(0)
← drawer_drop_can(0)
→ on_drawer_filled(1)
→ on_drawer_dropped(1)
← drawer_drop_can(1)
→ on_cash_inserted(200)
→ on_cash_inserted(100)
→ on_cash_inserted(50)
→ on_cash_filled(200)
→ on_cash_filled(100)
→ on_cash_filled(50)
← cash_drop_coin(200)
← cash_drop_coin(100)
← cash_drop_coin(50)
→ on_bin_fetch_all
→ on_bin_entry

Overview

Vending machine is a piece of hardware containing sensors and actuators for its components.

- ***Sensors*** register events from hardware components in the machine
  Sensors are therefore input for the software and output for the hardware. The control software can subscribe to events from specific sensors.
- ***Actuators*** pass information to the hardware components
  Actuators are therefore output for the software and input for the hardware.



Development

Ruby Code organization

`lib/` controller code

`test/` tests for controller code

`features/` story tests and steps

To deploy your controller code to the simulator, execute `rake deploy`

To run your tests, execute `rake` (or `autotest` to run them automatically).

To run the story tests, execute `rake features`

C# Code organization

The solution consists of 5 projects:

Project	Description
VendingMachine.Controller	Contains the control software that you need to write. The *Adapter* directory contains the sensor collection and the actuator collection that control should use to respond to sensor events and initiate actuator events
VendingMachine.Controller.unittests	Unit tests for the control software.
VendingMachine.Controller.specs	Feature tests for the control software
VendingMachine.Simulator	Contains hardware components simulating the hardware (directory *Hardware*) and the simulator GUI. This is also the startup project.
VendingMachine.Simulator.unittests	Unit tests for the hardware components

Where's the main?

Ruby:

Your objects controlling the machine are loaded by the executable **lib/main.rb.** An example:

```
require "vmlog"
require 'devices'

module Control
  include VMLog

  def self.main
    # here, you should load your vending machine control software

# as an example of using the actuators and sensors (seedoc/actuators_and_sensors)
    Devices.display_show(0, 'Choose')
    Devices.display_show(1, 'Please...')

    Devices.on_button_pressed(0) do
      log "cola pressed"

      Devices.display_show(0, 'Cola')
    end
  end
end
```

You can use the `log` function from the VMLog module to print message to stderr.

C#:

Your starting point for the control software is the **VendingMachineController** class, which has an **OnLoad** method that is called on startup:

```
public class VendingMachineController
{
    public void OnLoad()
    {
      // here, your vending machine control software will be loaded
    }
}
```

Subscribe to a sensor event

Ruby:

```
      Devices.on_button_pressed(1) do
          # put your event handling code here
      end
```

C#:

```
      SensorCollection.Instance().On(":button_press_0", delegate() {
          this.ButtonPressed();
      });
```

Fire an actuator

Ruby:

```
Devices.display_show(1, "test message")

Devices.drawer_drop_can(1)
```

C#:

```
ActuatorCollection.Instance().Fire(":drawer_drop_can_3");

ActuatorCollection.Instance().Fire(":display_show",
      new Object[] {0, "credits: 0"});
```

Hardware components

Buttons

Buttons register presses and contain sensors to notify software. The system contains the following buttons:

button_no	description
0	The cola button
1	The fanta button
2	The sprite button
3	The sisi button

Sensors

- on_button_pressed(<button_no>)
  fires when a button is pressed
  <button_no> is the number of the button

Drawers

Drawers contains cans of beverage. Drawers can drop cans into the bin. They contain actuators to drop a can and sensors to detect that a can is added to the drawer (*fill*) and to detect that a can has been dropped (*drop*). The system contains the following drawers:

drawer_no	description
0	The cola drawer
1	The fanta drawer
2	The sprite drawer
3	The sisi drawer

Sensors

- on_drawer_dropped(<drawer_no>)
  fires on dropping a can
  <drawer_no> is the number of the drawer
- on_drawer_filled(<drawer_no>)
  fires on putting a new can in the drawer
  <drawer_no> is the number of the drawer

Actuators

- drawer_drop_can(<drawer_no>)
  tells a drawer to drop a can
  <drawer_no> is the number of the drawer

Display

The display can show messages to the user. It contains two lines. The system has only one display.

Actuators

- display_show (<line_no>, <message>)
  tells the display to show a message
  <line_no> is the number of the line (0 based)
  <message> is the message to display

Cash register

The cash register can accept coins and drop them into the bin (to give money back). The cash register contains actuators to drop different types of coins and sensors to notify that a coin is inserted into the cash register. It recognizes the following coins:

coin_type	description
50	50 cents
100	1 euro
200	2 euro

Sensors

- on_cash_filled(<coin_type>)
  fires on filling a coin in the cash register's stock
  <coin_type> is the type of coin added
- on_cash_inserted(<coin_type>)
  fires on inserting a coin in the cash register as a paying action
  <coin_type> is the type of coin added

Actuators

- cash_drop_coin(<coin_type>)
  tells cash_register to drop a coin in the bin as a change return action
  <coin_type> is the type of coin to drop (50, 100, 200)

Bin

The bin catches cans and change. It contain sensors to detect cans or coins entering the bin and to detect that the bin is emptied,

Sensors

- on_bin_entry
  fires on something (a coin or a can) falling in the bin
- on_bin_fetch_all
  fires on something (a coin or a can) fetched from the bin by a user.