Smart Green House

Alexander Platz
Aravinth, S. Panchadcharam
Florian David Roubal
Martin Kessel
Sven Erik Jeroschewski

Telecommunication Networks Group
Technische Universität Berlin

Agenda

- **Motivation**
- **Communication**
- **System Design**
  - Sensor Node & Sensors
  - Actuator Node & Actuators
  - Control Center with User Interface
- **Lessons learned**
- **Future Work**
- **Live demo**

Motivation

You live far away from home and no one is there to take care of your plants?

You are a busy student and you don’t have enough time to water your plants?

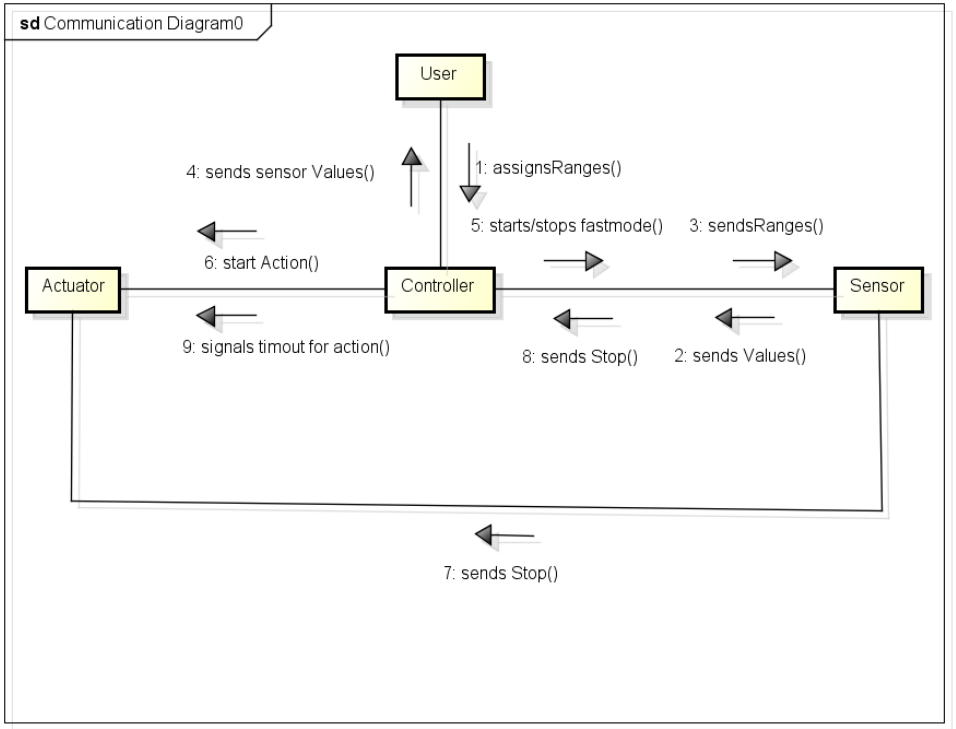
Our Smart Green House can help you!

Work Split Up

- **Sensor Node & Sensors**
  - Martin Kessel
- **Communication**
  - Sven Erik Jeroschewski
- **Actuator Node & Actuators**
  - Florian David Roubal
- **Control Center & UI**
  - Alexander Platz
  - Aravinth, S. Panchadcharam

Communication

Communication



Controller node sending message types

- **b message “send range”**
  - contains new range for sensor
  - sprintf(data, "b%d;%d;%d;%d;%d;%d;%d;%d", tempRange[0], tempRange[1], humRange[0], humRange[1], lightRange[0], lightRange[1], moistRange[0], moistRange[1]);

- **e message “set action” (actionType==1)**
  - assigns new action to the actuator
  - sprintf(data, "e%d;%d;%d;%d;", valveValue, valveTimeout, lightValue, lightTimeout);

- **h message “modFastMode” (actionType==2 or 3)**
  - turns fast mode in sensor on (1) or off (0) and transmits critical moist value
  - sprintf(data, "h%d;%d;", fastMode, criticalMoistValue);

Sensor node sending message types

- **a message**
  - contains sampled values
  - sprintf(data, "a%d;%d;%d;%d;", temperature, humidity, light, soil_moisture
- **j message**
  - signals instant off to actuator and control center
  - no data
  - sent when the current soil moisture threshold is reached

System Design

Sensor Node & Sensors

Sensor Implementation

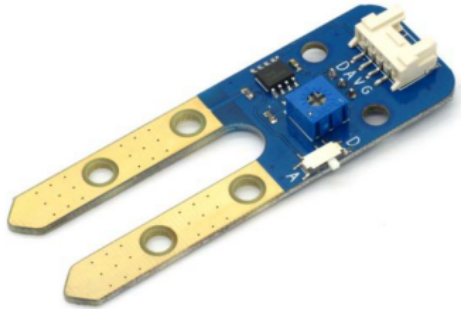
- Gather information about Waspmote Hardware
    - Which operating voltage? (3,3 V)
    - Maximal sinkable currents? (40 mA max per pin)
    - supported serial protocols? (I^2C, one Wire)

- Adapt Arduino libraries to Waspmote
    - Different includes
    - Partially different/missing functions (-> micros())
    - Waspmote API and IDE are buggy in our version

- Implement sensor functionality in node logic

Soil Moisture sensor

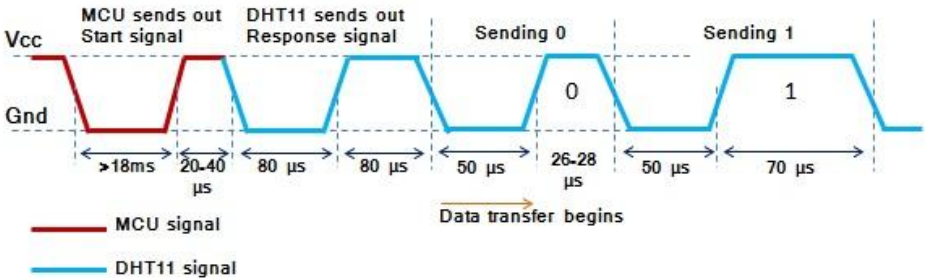
```
//Enable 3,3V Output Pin for moisture and light sensor
pinMode(SENS_PW_3V3,OUTPUT);
digitalWrite(SENS_PW_3V3,HIGH);

 //Get moisture value
 moistVal = analogRead(ANALOG1);

 moistVal < 400 means very wet
```

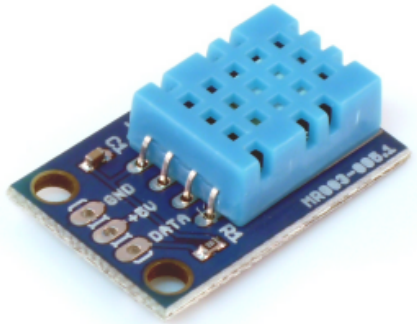


Parameter	Min.	Typical	Max.	Unit
Working voltage	2.1	5	5.5	VDC
Analog output voltage (VCC=5V)	0	Vout	5	V
Digital output voltage (VCC=5V)	0	-	5	V
Working current (VCC=5V)	-	5	-	mA
Threshold hysteresis ΔUth	-	VCC*0.09	-	V

DHT11 temperature and humidity sensor

- Waspmote API doesn´t support counting microseconds
  - simple counter implemented by using loop with constant cycles and counting repetitions

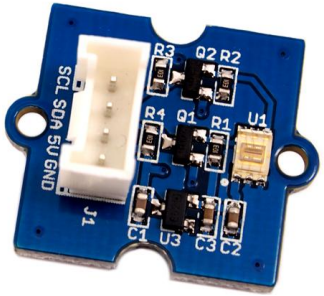


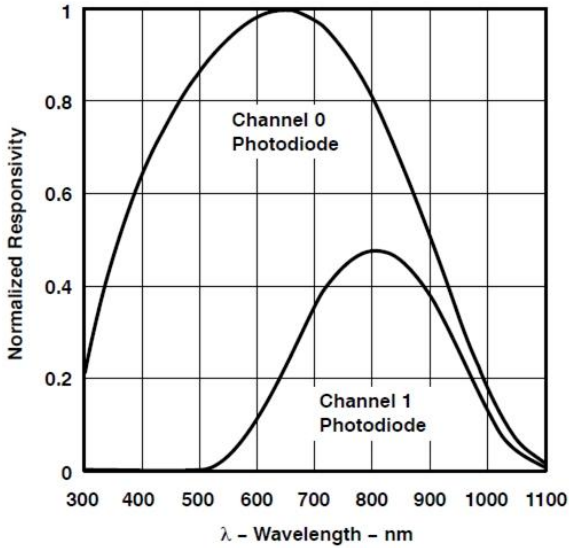


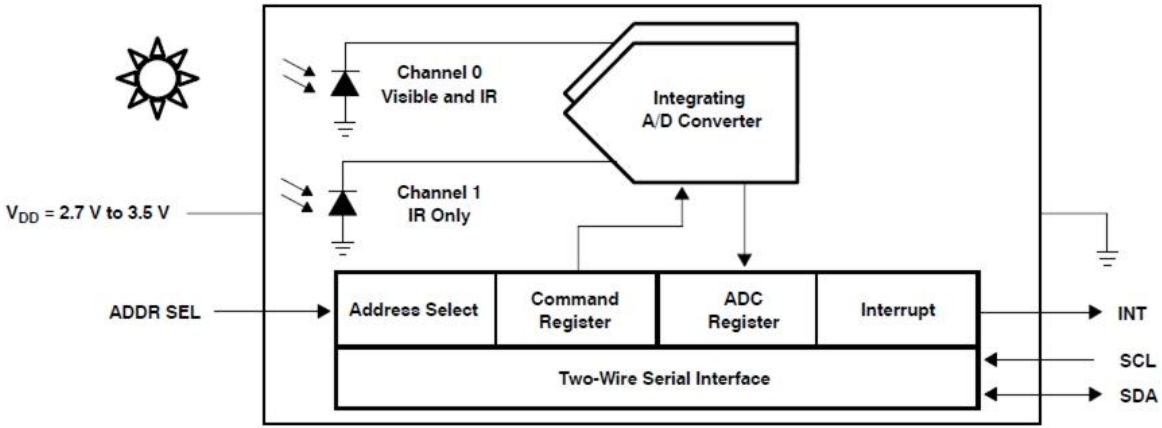
Supply voltage	+5V
Supply current (running)	0.5mA typ. (2.5mA max.)
Supply current (stand-by)	100uA typ. (150uA max.)
Temperature range	0 / +50°C ±2°C
Humidity range	20-90%RH ±5%RH
Interface	Digital

Grove Digital Light sensor (TSL2561)

- Connection was simple because of Waspmote supporting the Two-Wire Interface and a library for Arduino was available







Battery life of Sensor Node estimation

Battery capacity: 2300 mAh

Active components sink:

- **moisture sensor: 5 mA**
- **temp & humidity: 0,5 mA**
- **light sensor: 0,25 mA**
- **Xbee 37 - 64 mA**
- **Waspmote (ON) 9 mA**

Let´s assume node samples and sends every hour -> 24 times per day

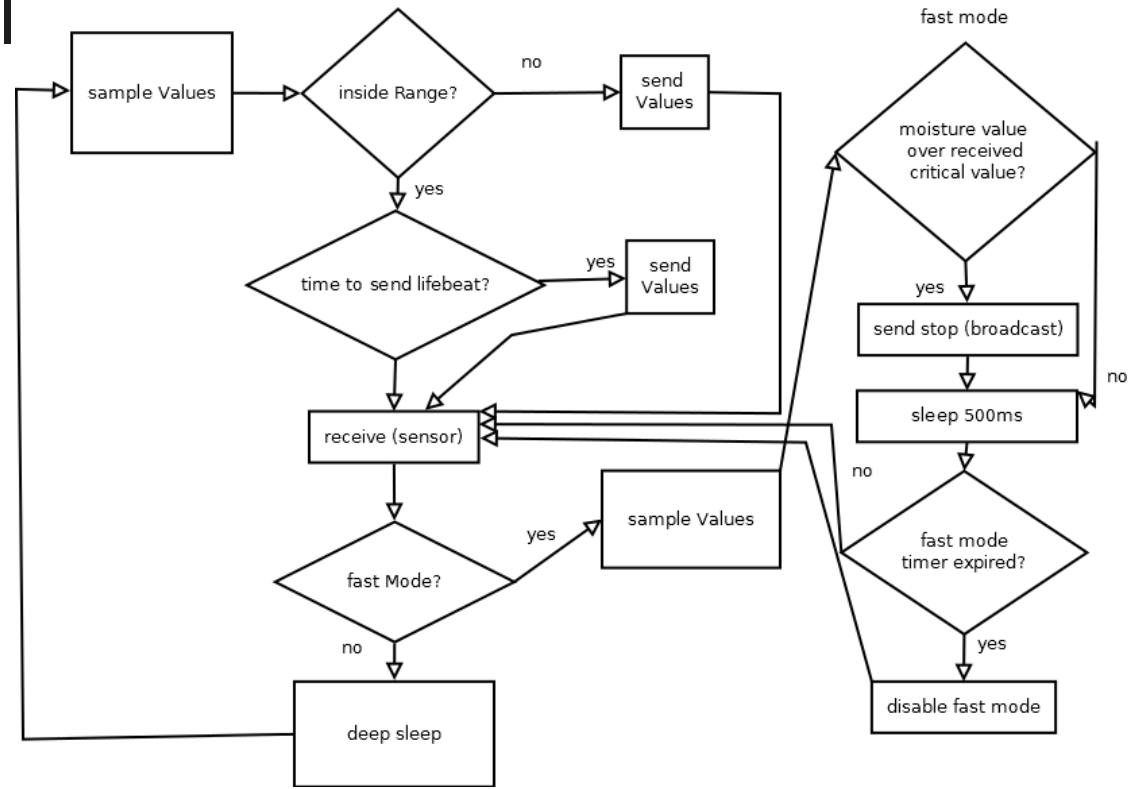
- **1 time needs (very roughly) 60 mA for 20 seconds**
  - **-> 60 mA 4800 s -> 60 mAh**

-> 2300 mAh / 60 mAh = 38 days

Sensor node sending message types

- **a message**
  - contains sampled values
  - sprintf(data, "a%d;%d;%d;%d;", temperature, humidity, light, soil_moisture)
- **j message**
  - signals instant off to actuator and control center
  - no data
  - sent when the current soil moisture threshold is reached

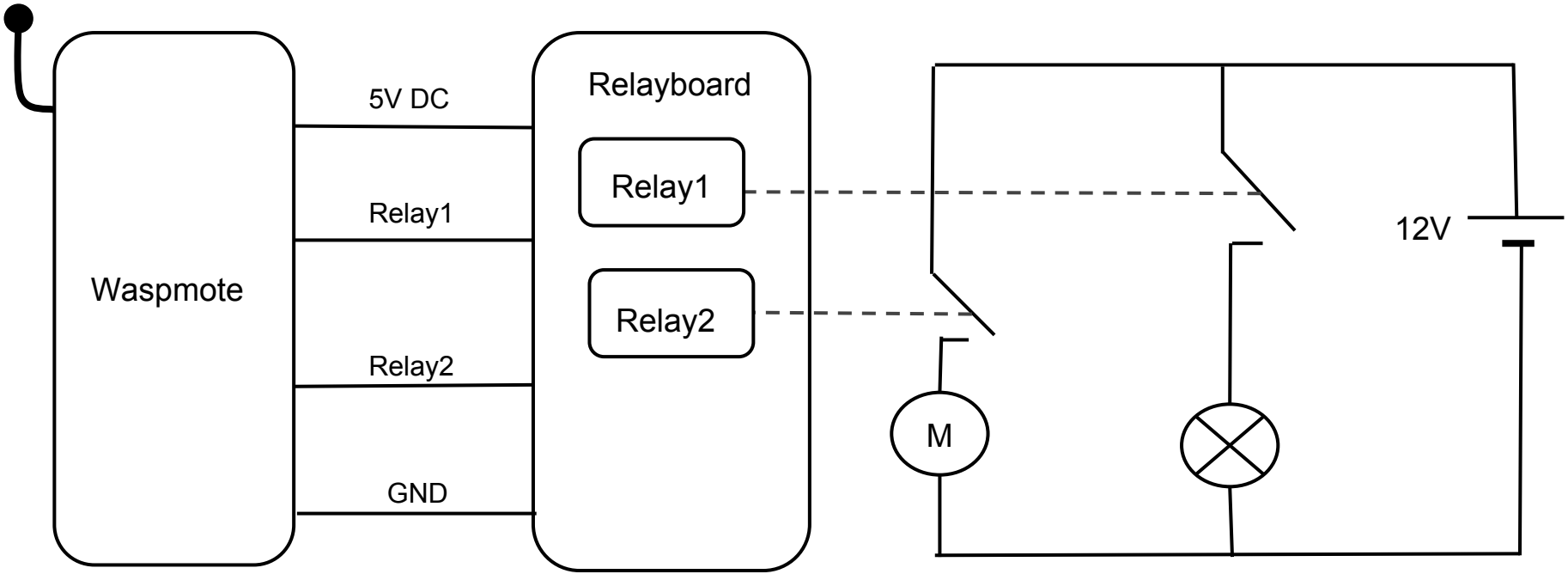
sensor node control logic



System Design

Actuator Node & Actuators

- **Hardware Components**



Actuator node & actuator

- **Relay Shield**

Power Supply: 5 V

- Switch off the power supply to save energy

2 Channels

- 2 actuators can be supported

- **Water Pump**

Measured Output:

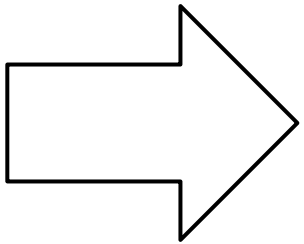
1 Liter in 7 Seconds
100ml in 0.7 Seconds

- For not pouring too much: good timing for the pump is required
- To save the plants for too much water: Alarm-Off message is enabled between sensor node and actuator

Actuator node & actuator

- **Light for Plants**

249€ at amazon



0.60€ at Segor

- **Features**

- **Run and stop both actuators independently**
  - It is possible run the pump while receiving a new message for the light (and vice versa)
  - realised with timers

- **Alarm OFF**
  - If water pump pumps too much water into the flowerpot the sensor sends an alarm off
  - the pump is shut down immediately
  - the flower pot never gets too wet

- **Features II**

- **Powersaving mode**
  - if no actuator is needed the whole relay board will be shut down

- **2 programmable LED show the on or off mode of both actuators**

- **Checking correctness of received message**
  - If a wrong entry for valveValue or lightValue was received the red LED will blink and a message will be returned to the serial port

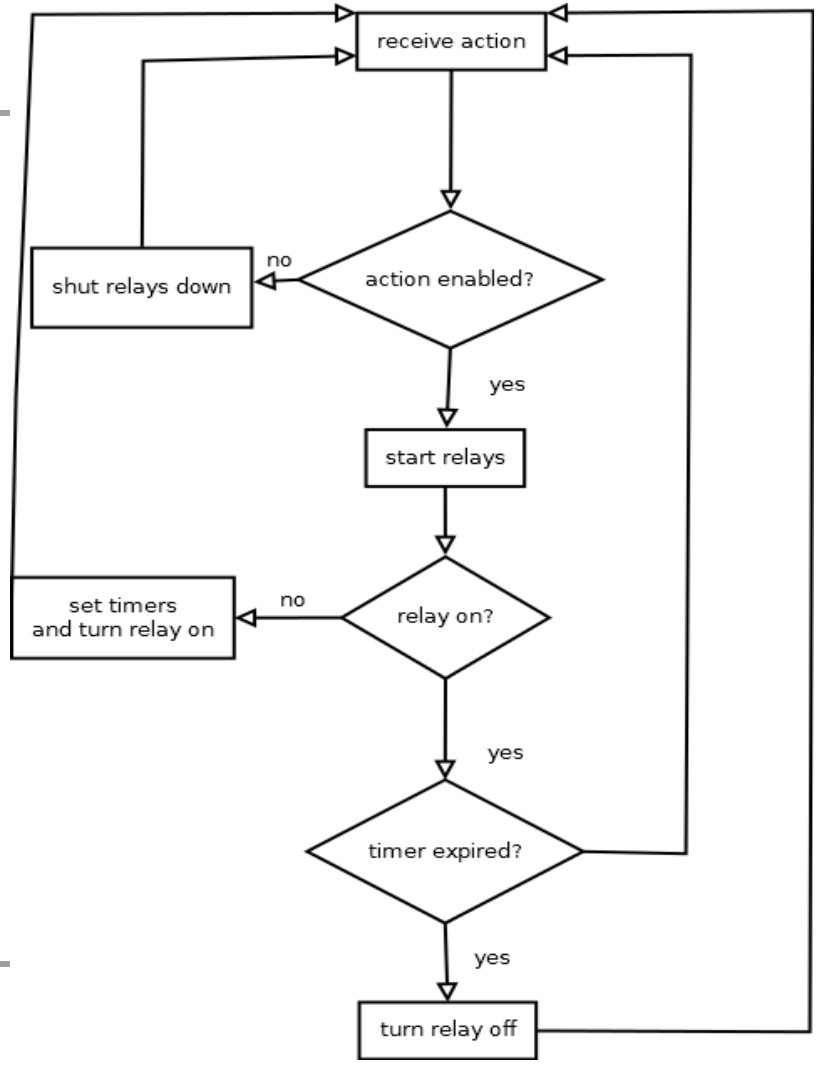
- **new received message can overwrite the actual value**
  - thus the system is very flexible if new computation was done

actuator node control logic

Flow Example

- receive message
- activate relay
- set timer
- receive message
- check if timer expired
- receive message
- timer1 expired: turn off
- As far as BOTH timer are expired the relay board is down

BUT: If a alarm-OFF message arrives: The Pump is turned off immediately



Message Types for Actuator

- **e message**
  - standard message type to activate actuator for a certain time
  - both actuators can be activated separately (but don't have to)
  - e1;2;1;5
  - e valveValue; valveTimeout; lightValue; lightTimeout
  - if valveValue = 1, set timer1 to valveTimeout
  - if lightValue = 1, set timer2 to lightTimeout

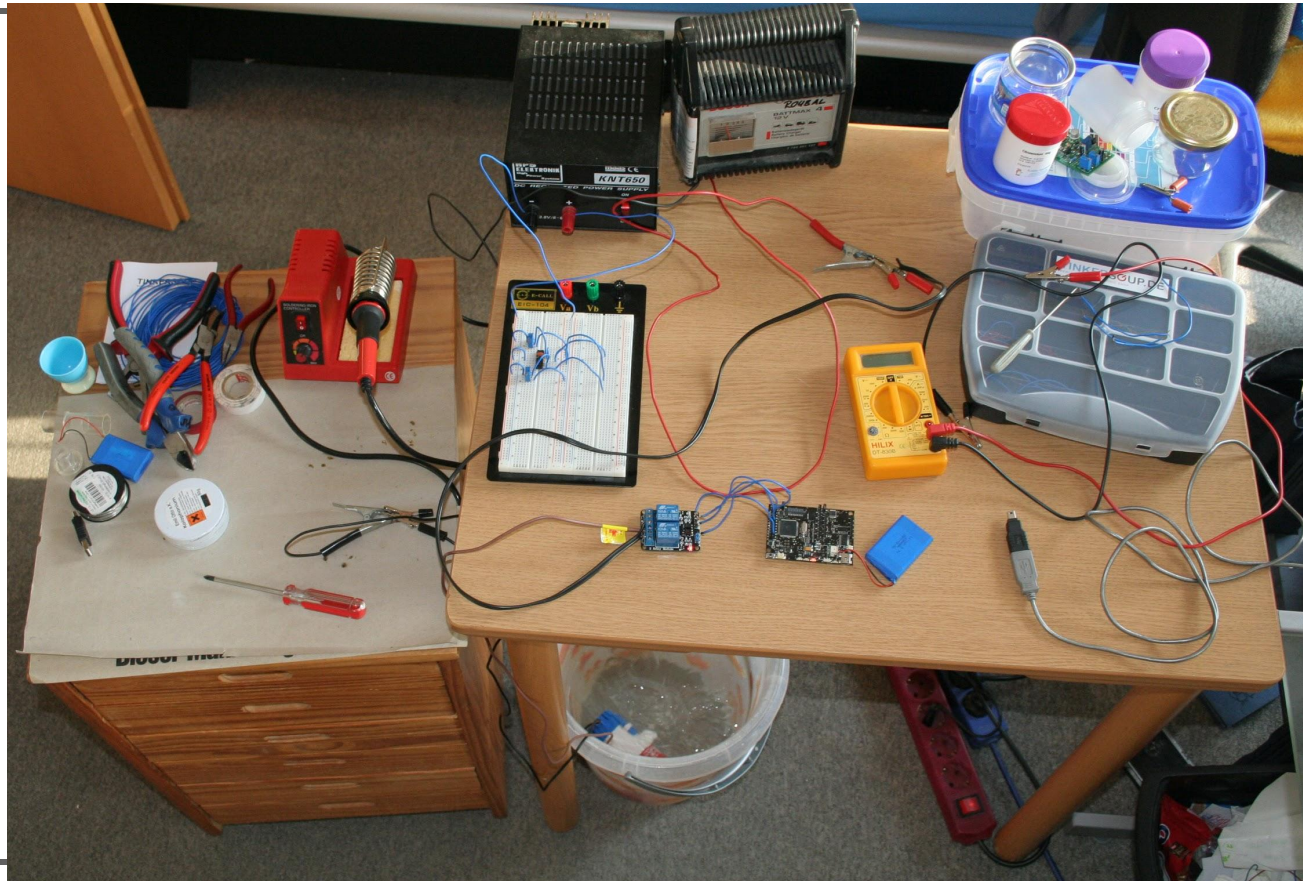
- **j message**
  - Alarm off message
  - if j was received pump is turned off immediately

Actuator node & actuator

Florian David Roubal

Look behind the scenes:

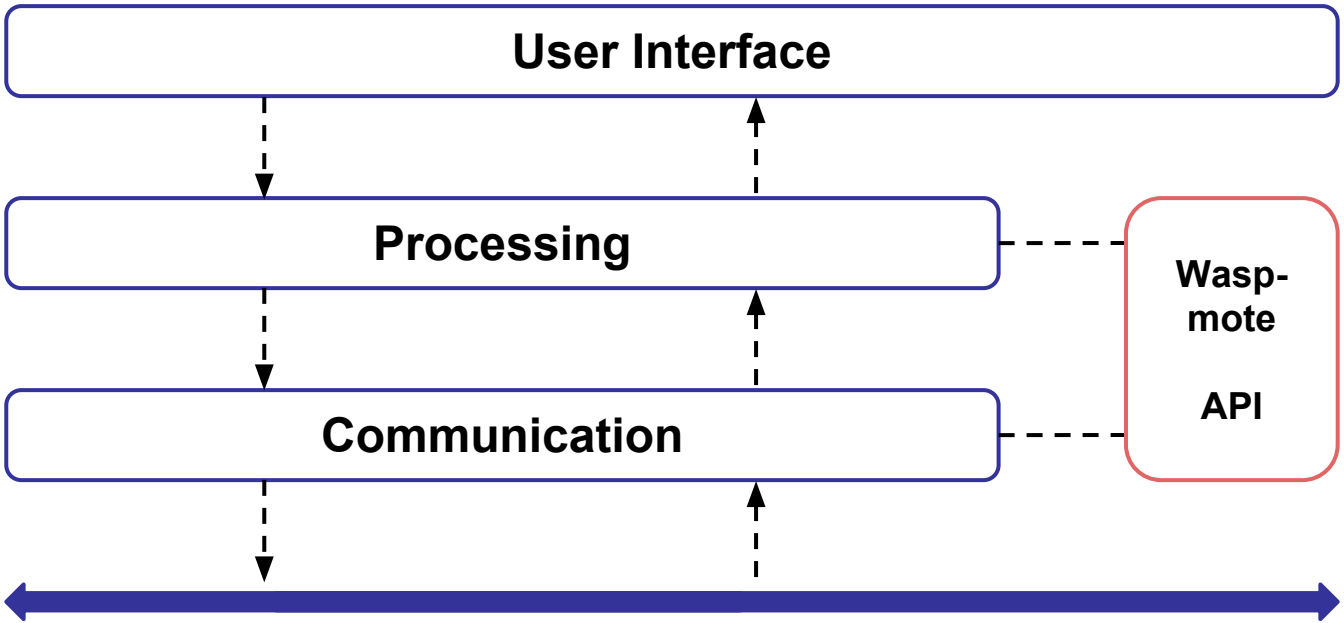
An Engineer's laboratory :-)



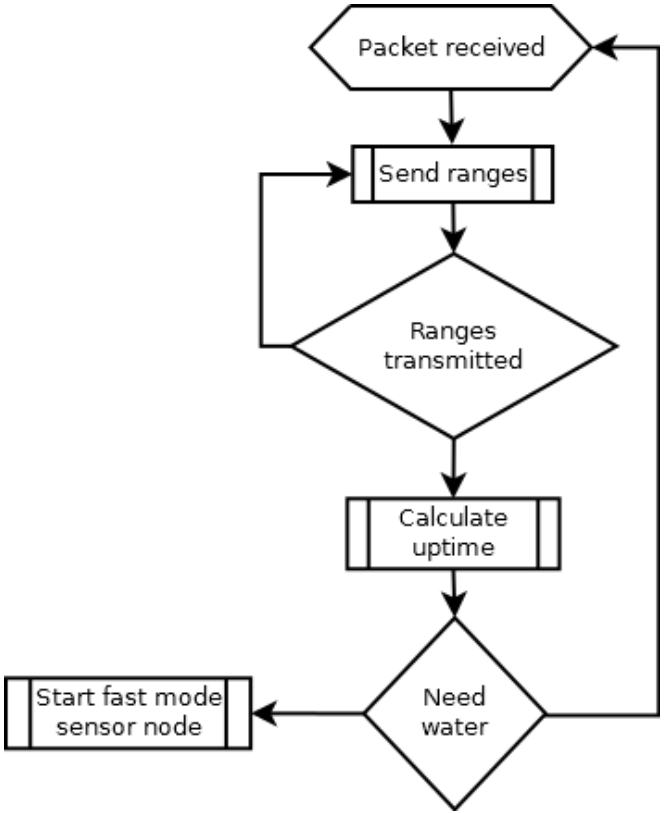
System Design

Control Center with User Interface

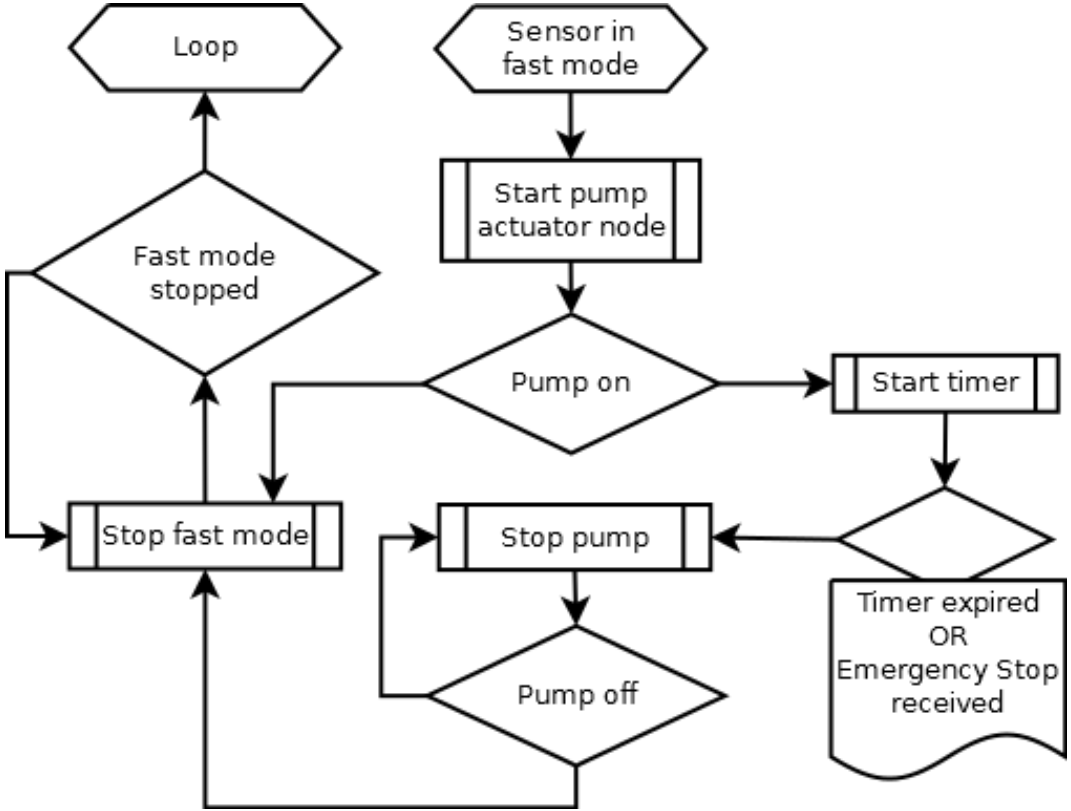
Functional layers



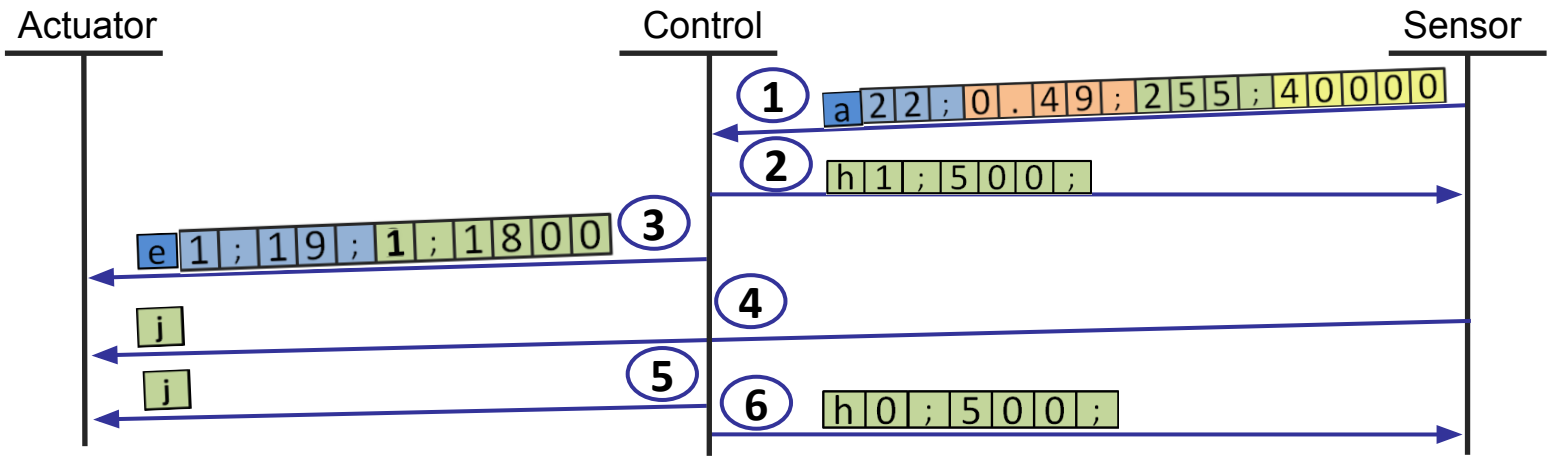
Design - Processing sensor data



Design - Processing fast mode



Design - Communication



1. Send current sensor data
2. Switch to fast mode
3. Switch-on pump
4. Switch-off pump
5. Emergency stop pump
6. Switch-off fast mode

Implementation - stop pump with retry (1/2)

```
// --- When timer expires or emergency stop received
// --- switch-off the valve in actuator node --> sendAction(4, x)
int stopPump(){
  int sendingStatus = sendAction(4, 0);
  if(sendingStatus == 1) {
        for(int retries = 0; retries <= NOSENDRETRIES; retries++) {
              XBee.println("Trying again... ");
              delay(DELAY);
              sendingStatus = sendAction(4, 0);
              if(sendingStatus == 0) {
                    actionOn = false;
                    xbee868.flush();
                    return sendingStatus;
              }
         }
         XBee.println("Control Center -> Actuator Node: Stop pump not successful" );
         return sendingStatus;
  }
  else {
       actionOn = false;
       return sendingStatus;
  }
}
```

Implementation - stop pump with retry (2/2)

```
int sendAction(int actionType, int criticalMoistValue) {
  int status = 0;
  initXbeePacket();
  ...
  // --- OFF-Signal valve actuator node
  else if(actionType == 4) {
       sprintf(data, "j");
       xbee868.setDestinationParams(paq_sent, "0013A2004078407E", data, MAC_TYPE, DATA_ABSOLUTE);
  }
  xbee868.sendXBee(paq_sent);
  if( !xbee868.error_TX ) {
       XBee.println("Control Center: Action sent.");
  }
  else {
        XBee.println("Control Center: Error while sending action.");
        status = 1;
  }
  free(paq_sent);
  paq_sent=NULL;
  return status;
}
```

User Interface

- User Interface enables the user to configure, control and monitor Smart Green House
- It is a platform independent program that runs in any modern web browser
- It is implemented with various technologies such as Node.JS, Websockets, Serial Port interfaces, HTML, Javascript, etc

Communication between UI and CC

- User Interface creates a Serial connection with the control center by accessing the UART port via USB cable
- Waspmote is implemented with Hardware and Software Serial, so that only available UART port is not locked by any program.
- Waspmote API has 2 Serial classes (Xbee.print & USB.print), as it uses UART port to communicate via Xbee

```
serialPort.write("S:UI:GET:DATA:E");
USB.print("S:CC:RANGE:15:30:30:50:20000:30000:400:600:E");
USB.print("S:CC:ACTUATOR:1:2005:9000:E");
USB.print("S:CC:SENSOR:1:15:E");

serialPort.write("S:UI:SET:RANGE:15:30:30:50:20000:30000:400:600:E");

USB.print("S:CC:RANGE:15:30:30:50:20000:30000:400:600:E");
```


Smart Green House
Air Temperature
30 °C
Humidity
50
Luminosity
30000
Soil Moisture
600
15 - 30
30 - 50
/dev/cu.usbmodemfd131
SAVE CHANGES
Pump is switched on at 2014-07-21 19:47:00 for 480 Seconds
Light is switched on at 2014-07-21 19:47:00 for 120 Seconds
Network Console
[2014-07-21 19:47:44.706] [Sensor] Humidity is 30
[2014-07-21 19:47:43.707] [Sensor] Air Temperature is 15
[2014-07-21 19:47:42.712] [Actuator] Light is on for 120 seconds
[2014-07-21 19:47:41.712] [Actuator] Pump is on for 480 seconds
[2014-07-21 19:47:40.713] [Sensor] Soil Moisture range is [400 : 600]
[2014-07-21 19:47:40.713] [Sensor] Luminosity range is [20000 : 30000]
[2014-07-21 19:47:40.713] [Sensor] Humidity range is [30 : 50]

Lessons learned

Lessons Learned

- **Typical Problems with waspmote**
  - know the platform/hardware before starting the project
  - detach the antenna for every flashing act
  - compared to other IDE's the compiling and flashing process takes a lot of time
  - battery cables can break very easily
  - string based communication scheme can be unhandy (transmitting structs or objects would be a better approach for our project)
  - stochastic packet losses inside the antenna range

⇨ Personally we do not recommend Waspmote for larger projects

Future Work

ideas

Future Work - Ideas

- Create statistics for error- and meteorological statistics
  - F.e. log light during the whole day
- Connect Waspmote to the Internet to include and process data of weather forecasts
- Outdoor function (don’t pour during the day)
- add scalability by adding dynamic addressing
- Scalability by adding persistence

Live demo

Thank you for your attention.