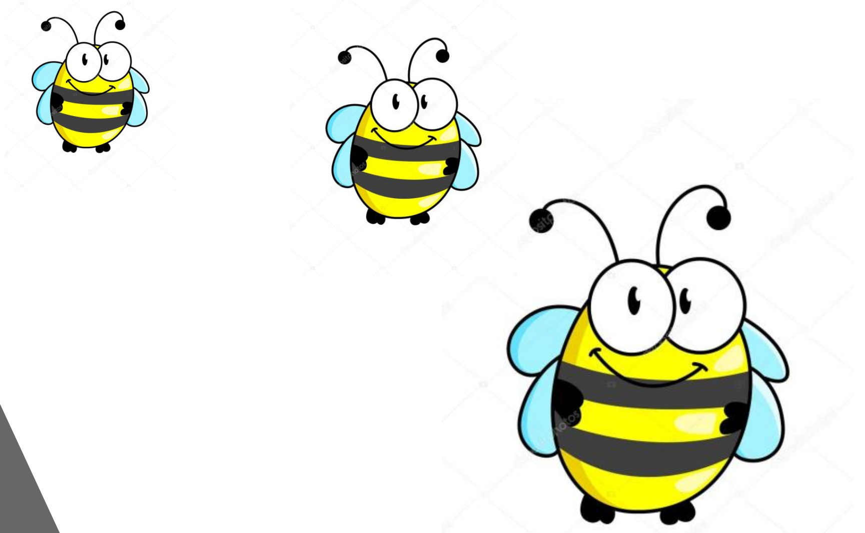
beeHouse-433

Smart **house** based on 433Mhz RF

Davide Neri
Email: davide.neri@di.unipi.it

Course: Reti Ad Hoc
A.a. 2016-2017



beeHouse-433: goals

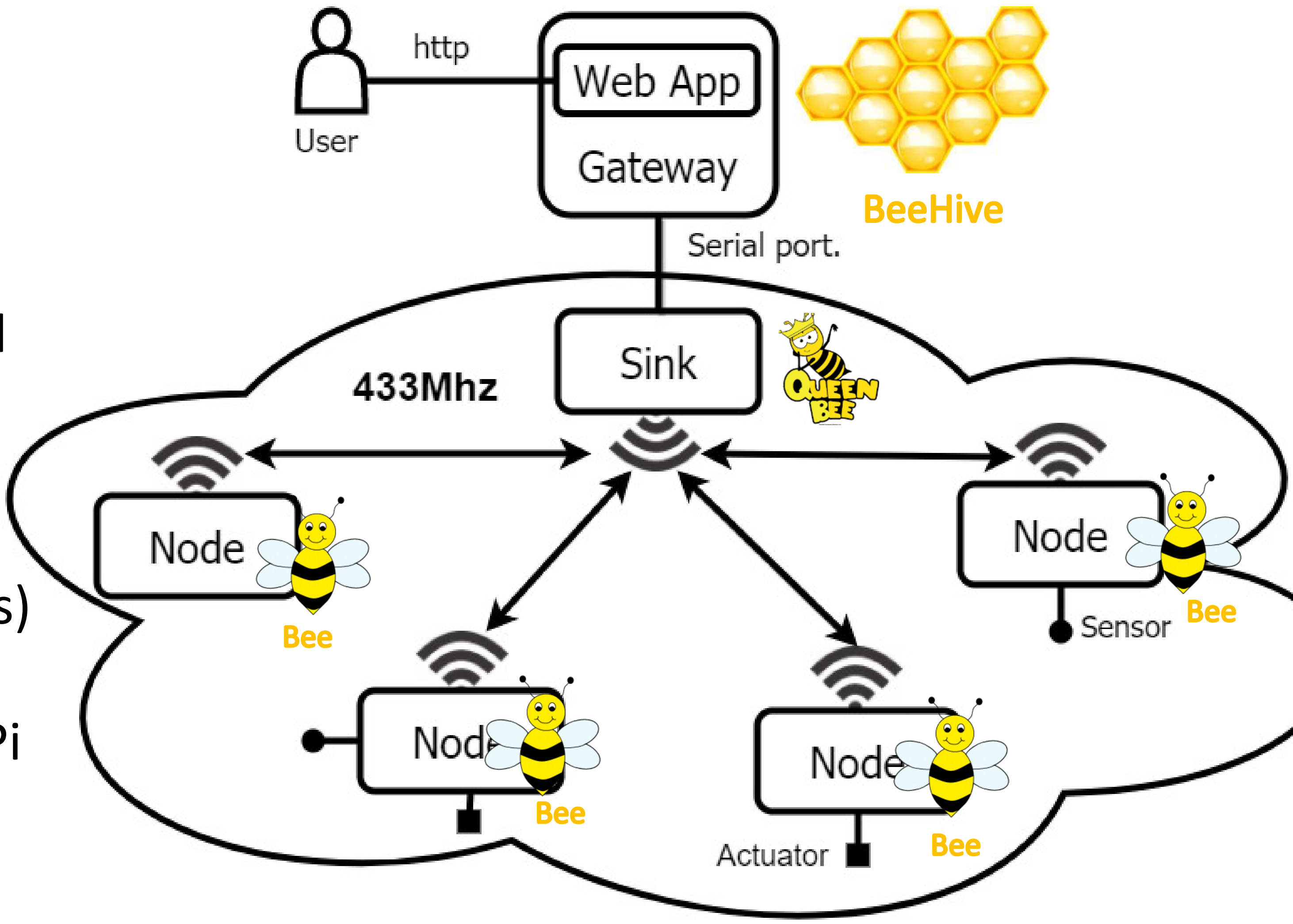
Monitor/Control/Automate your **house**.

Low cost wireless communication using **433Mhz** band

Organization likes **Bees** world…

System architecture

- Star topology (*sink* is the central node)
- Bidirectional communications based on 433Mhz RF.
- Components of architecture:
  - **Node(s)** (bees) – Arduino Nano(s)
  - **Sink** (beeQueen) – Arduino Uno
  - **Gateway** (BeeHive) - RaspberryPi



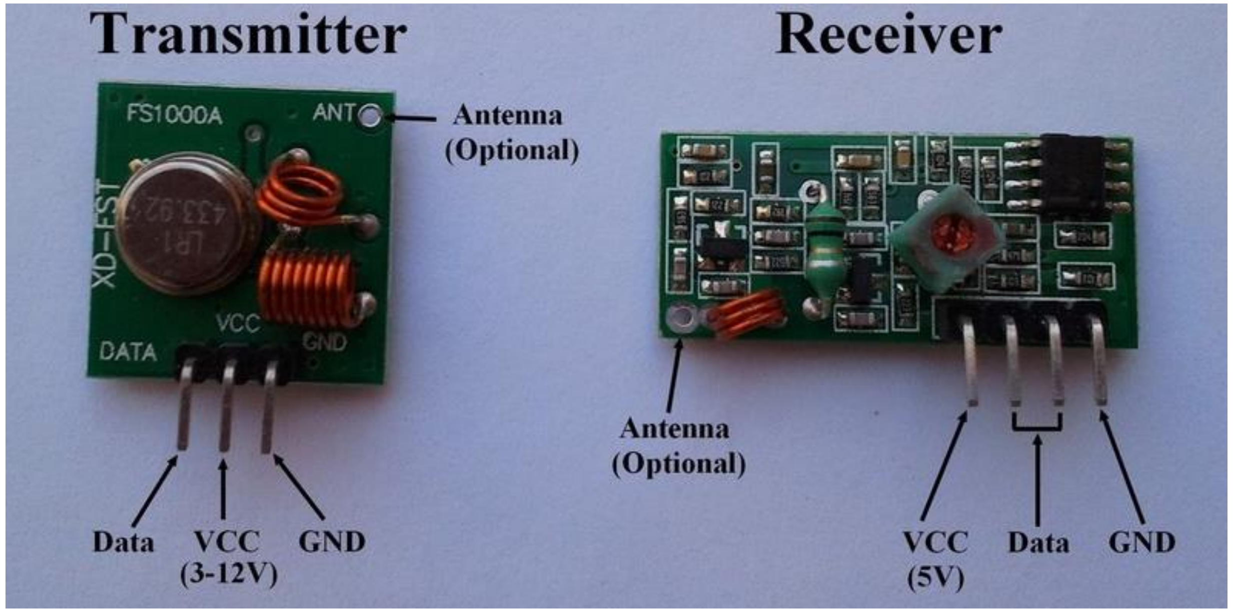
Wireless communication: 433Mhz transceivers

Specifications RF 433MHz ***Transmitter***

- Frequency Range: 433.92MHz
- Input Voltage: 3-12V
- Modulation: ASK
- Price: 1 – 2 euro

Specifications RF 433MHz ***Receiver***

- Frequency Range: 433.92 MHz
- Modulation: ASK
- Input Voltage: 5V
- Price: 1 – 2 euro



Range (with good antenna) up to hundred of meters.

Wireless Communication – 433 Mhz

Why choosing 433Mhz band instead of Wifi (2.4 GHz, 5Ghz) ?

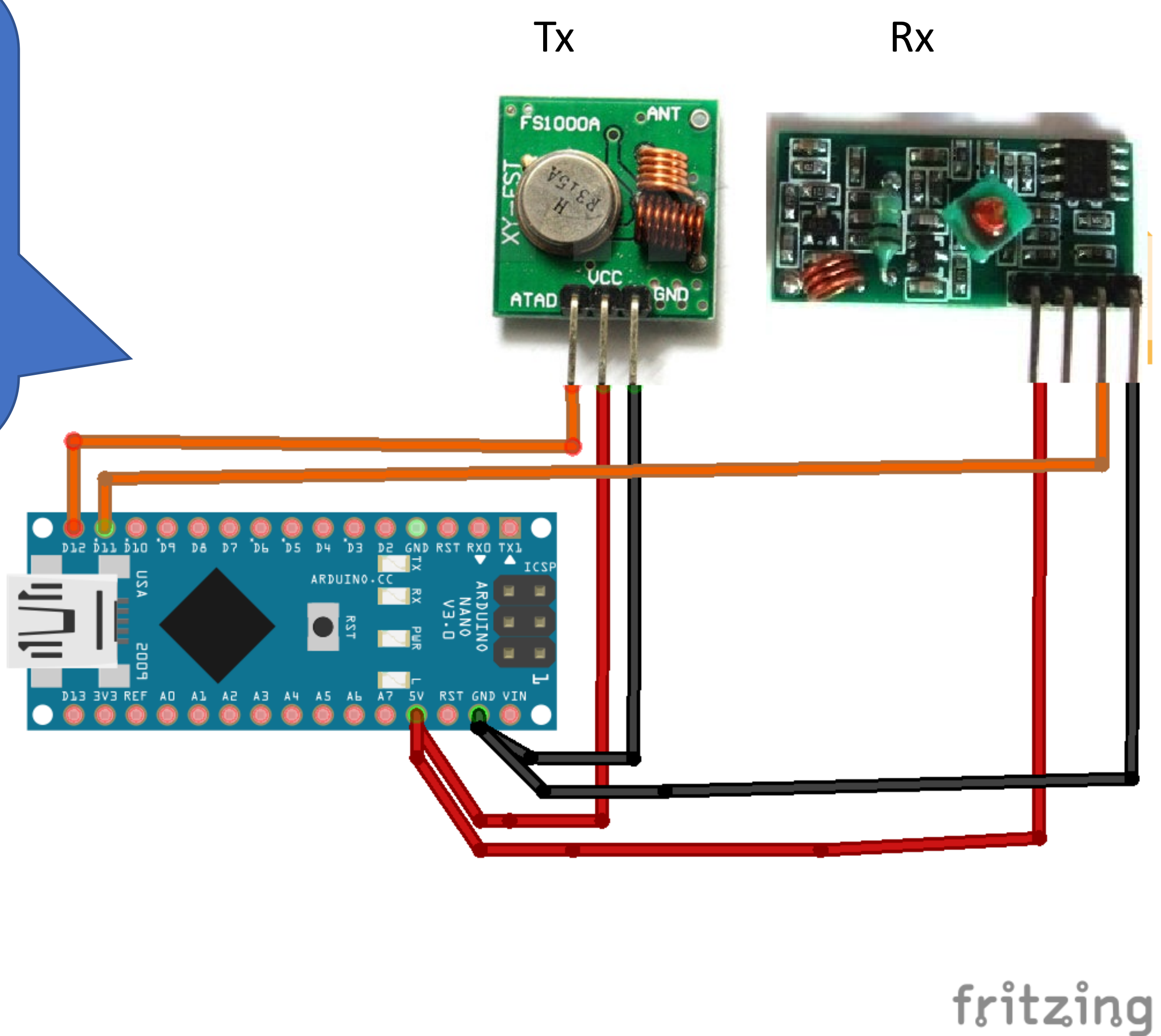
- **ISM** (industrial, scientific and medical) radio bands.
- Little interference with other networks (e.g., house wifi, mobiles phones)
- Cheap transceivers

Hardware Components – node(s)

Arduino Nano

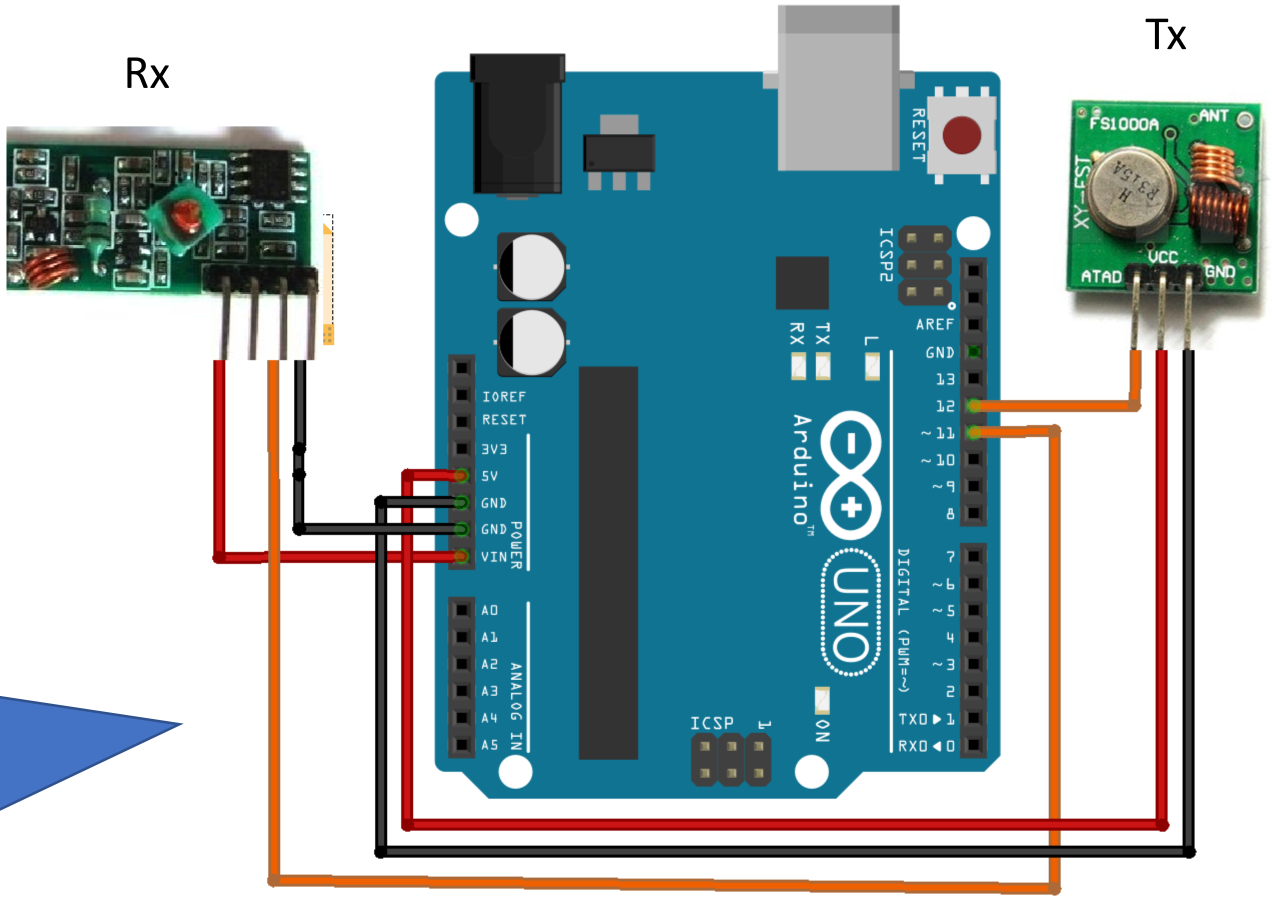
- Atmel ATmega328
- 16 MHz
- 4,31 x 1,85 cm
- 22 Digital pins

433Mhz Trasmitter and Receiver



Tx
Rx
FS1000A
ANT
XY-FST
ATAD
VCC
GND
D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RX0 TX1
D13 3V3 REF A0 A1 A2 A3 A4 A5 A6 A7 5V RST GND VIN
ARDUINO.CC
ARDUINO NANO V3.0
ICSP
fritzing

Hardware components - sink



Arduino UNO

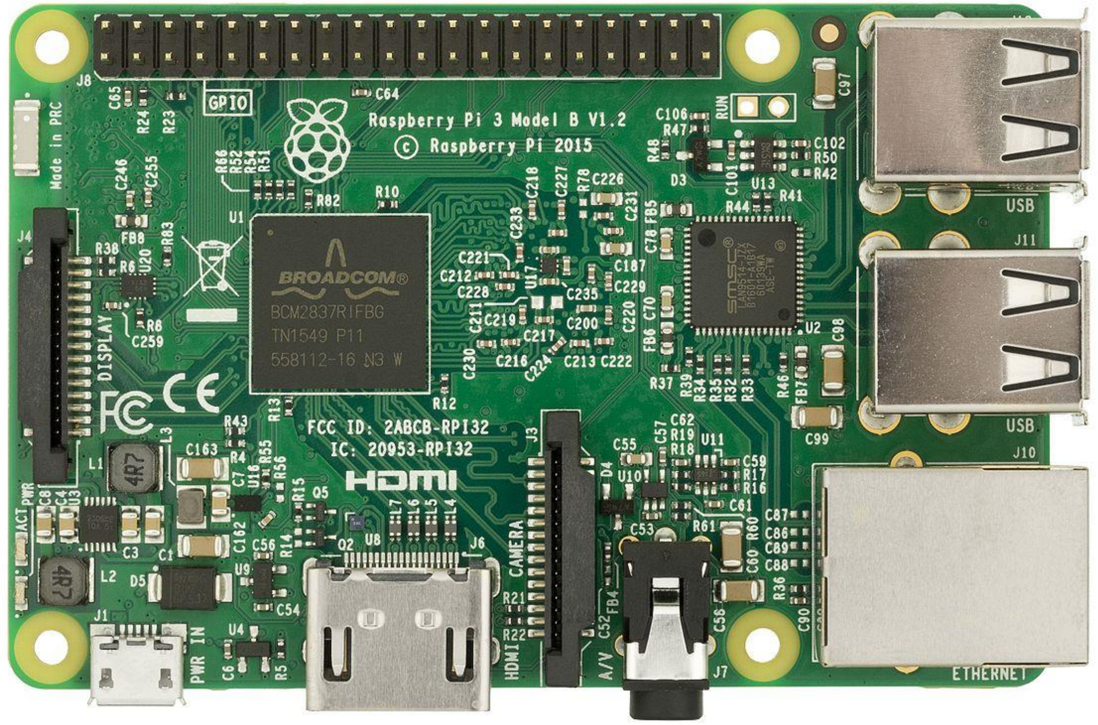
- Atmel ATmega328
- 16MHz
- 6.86 cm, 5.34 cm
- 14 Digital pins
- 6 Analog pins

433Mhz Trasmitter and receiver

Hardware components - Gateway

Raspberry Pi (v2)

- ARM1176JZF-S
- 900 MHz
- RAM 512 MB
- 17 GPIO pins (out of 26)
- USB Port
- Ethernet port



Softwares - 433 Mhz communication

RadioHead [1] is a Object oriented radio library for embedded microprocessors

BeeHouse-433 uses:
RF_ASK Driver
RHDatagram Manager:

Platform supported (some):

- Arduino and the Arduino ID
- Adafruit Feather
- Linux and OSX Using the RHutil/HardwareSerial class
- …

Offers 2 main sets of classes:

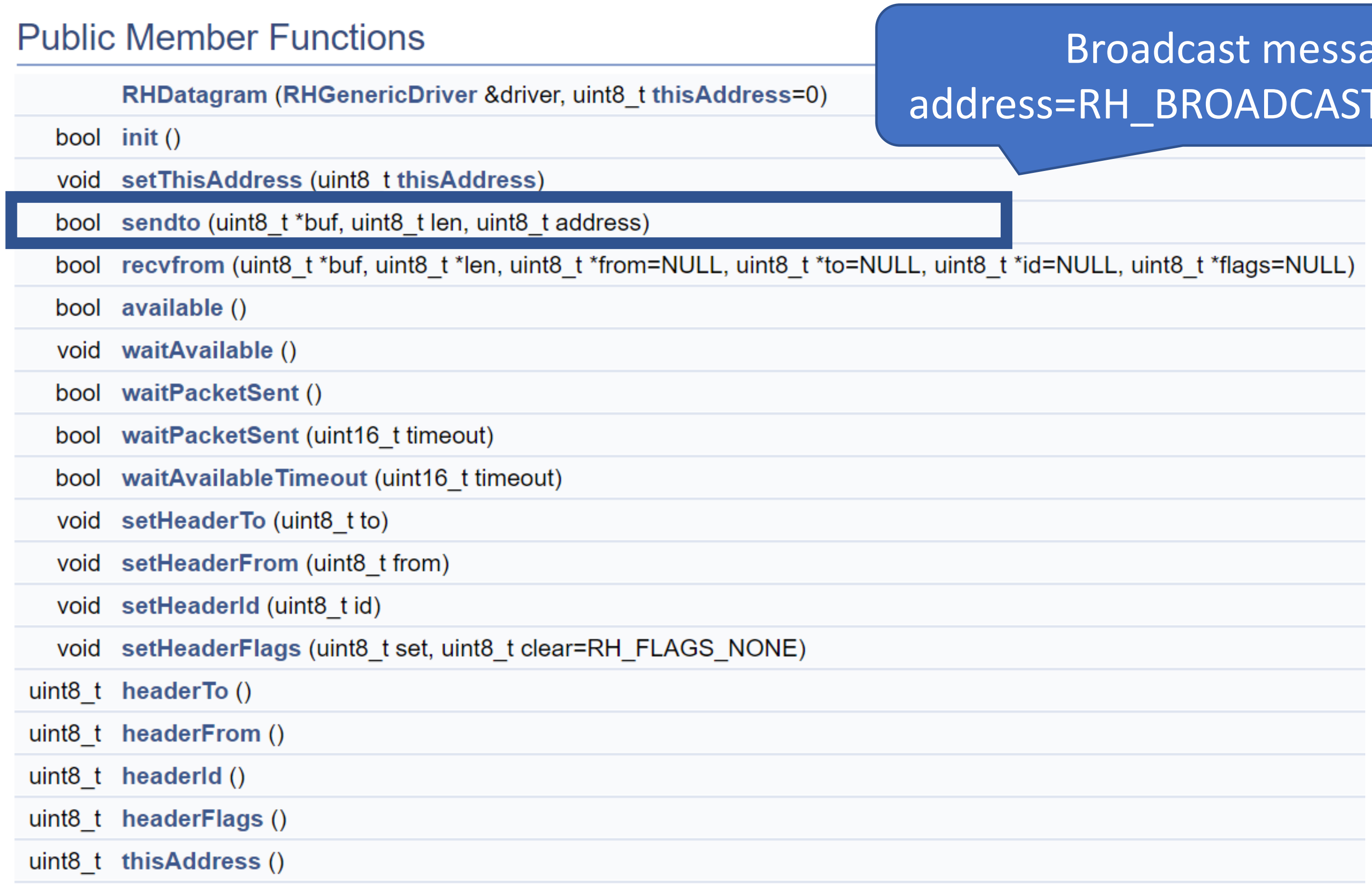
- **Drivers**: low level access to a range of different packet radios and other packetized message transports (**RF22** , **RF69**, ASK,…)
- **Mangers:** provide high level message sending and receiving facilities:
  - **RHDatagram** Addressed, unreliable variable length messages, with optional broadcast facilities.
  - **RHReliableDatagram** : Addressed, reliable, retransmitted, acknowledged variable length messages.
  - **RHRouter:** Multi-hop delivery from source node to destination node via 0 or more intermediate nodes, with manual routing.
  - **RHMESH:** Multi-hop delivery with automatic route discovery and rediscovery.

Any Manager may be used with any Driver, A Driver can be used without a Manager.

Other libraries*: VirtualWire(deprecated)* [3], *RCSwitch* [4], *Other* [5]

RadioHead library: RHDatagram manager

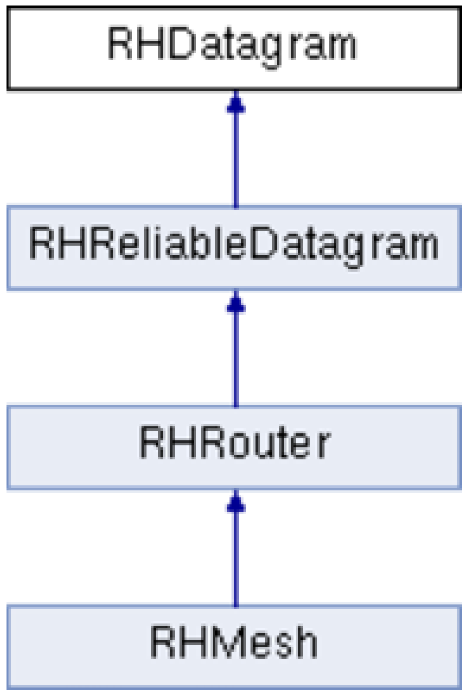
RHDatagram Addressed, unreliable variable length messages, with optional broadcast facilities.



Public Member Functions

	RHDatagram (RHGenericDriver &driver, uint8_t thisAddress=0)
bool	init ()
void	setThisAddress (uint8_t thisAddress)
bool	sendto (uint8_t *buf, uint8_t len, uint8_t address)
bool	recvfrom (uint8_t *buf, uint8_t *len, uint8_t *from=NULL, uint8_t *to=NULL, uint8_t *id=NULL, uint8_t *flags=NULL)
bool	available ()
void	waitAvailable ()
bool	waitPacketSent ()
bool	waitPacketSent (uint16_t timeout)
bool	waitAvailableTimeout (uint16_t timeout)
void	setHeaderTo (uint8_t to)
void	setHeaderFrom (uint8_t from)
void	setHeaderId (uint8_t id)
void	setHeaderFlags (uint8_t set, uint8_t clear=RH_FLAGS_NONE)
uint8_t	headerTo ()
uint8_t	headerFrom ()
uint8_t	headerId ()
uint8_t	headerFlags ()
uint8_t	thisAddress ()

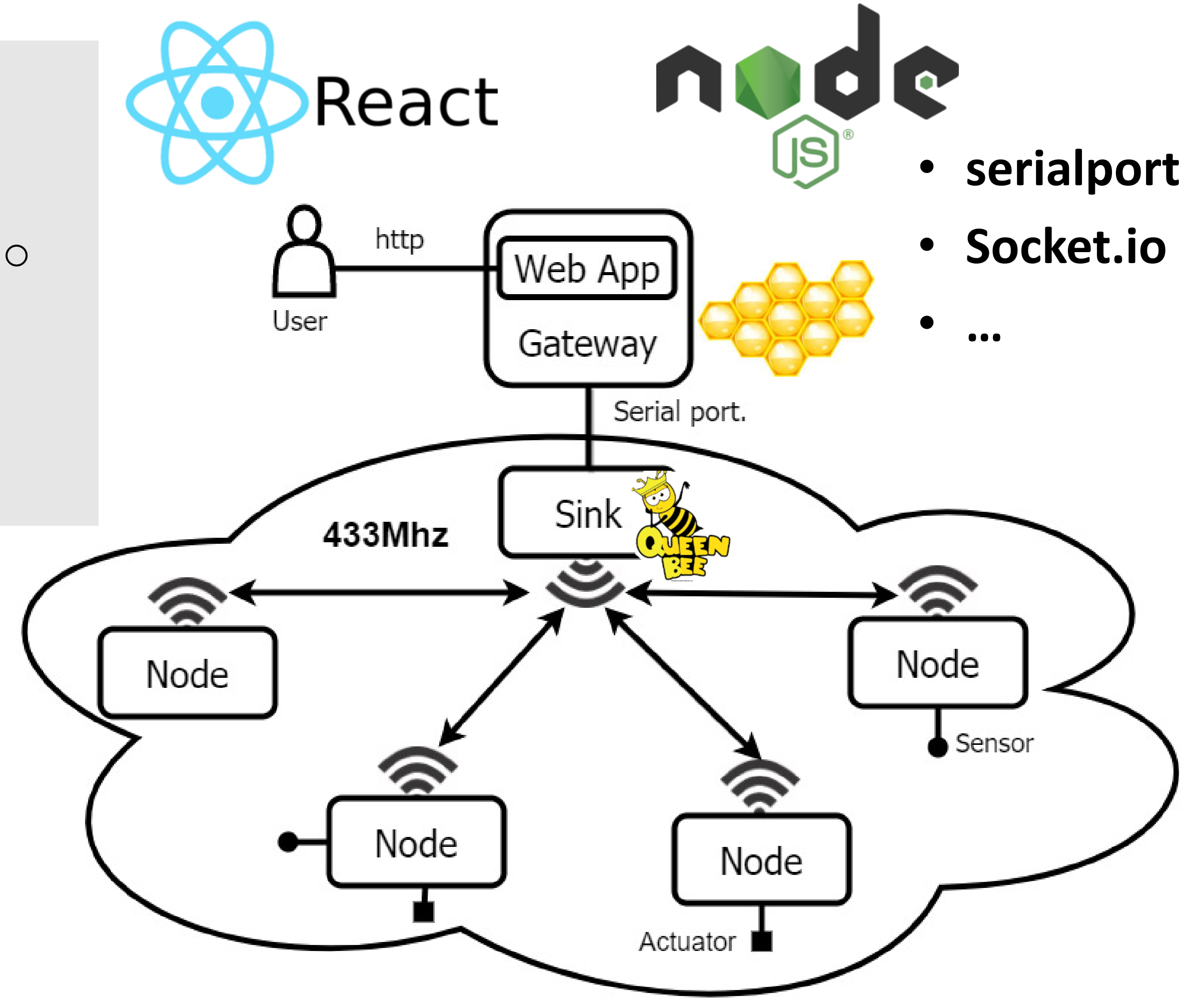
Broadcast message address=RH_BROADCAST_ADDRESS



Softwares - gateway (Beehive-server)

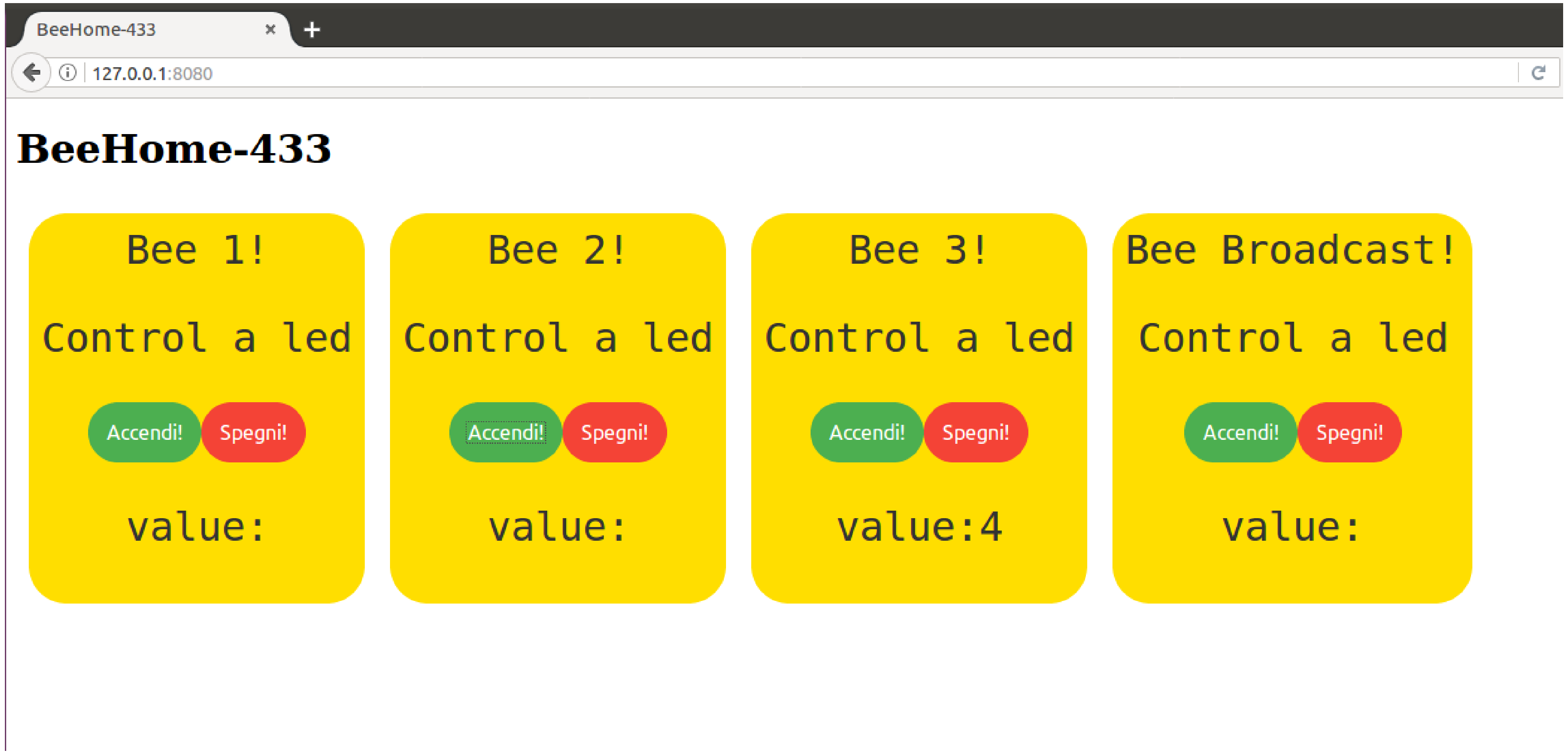
```
# code gateway (beehive-server)

m = Receive(msg) via serial port from sink
Send event("name", m ) to users via socket.io

m = receive event("name", msg) from user
Send(m) via serialport to sink
```

```
msg {
    src: Number,
    dst: Number,
    op: Number,
    data: Number
}
```



- **serialport**
- **Socket.io**
- **…**

WebApp interface



BeeHome-433

Bee 1!

Control a led

value:

Bee 2!

Control a led

value:

Bee 3!

Control a led

value:4

Bee Broadcast!

Control a led

value:

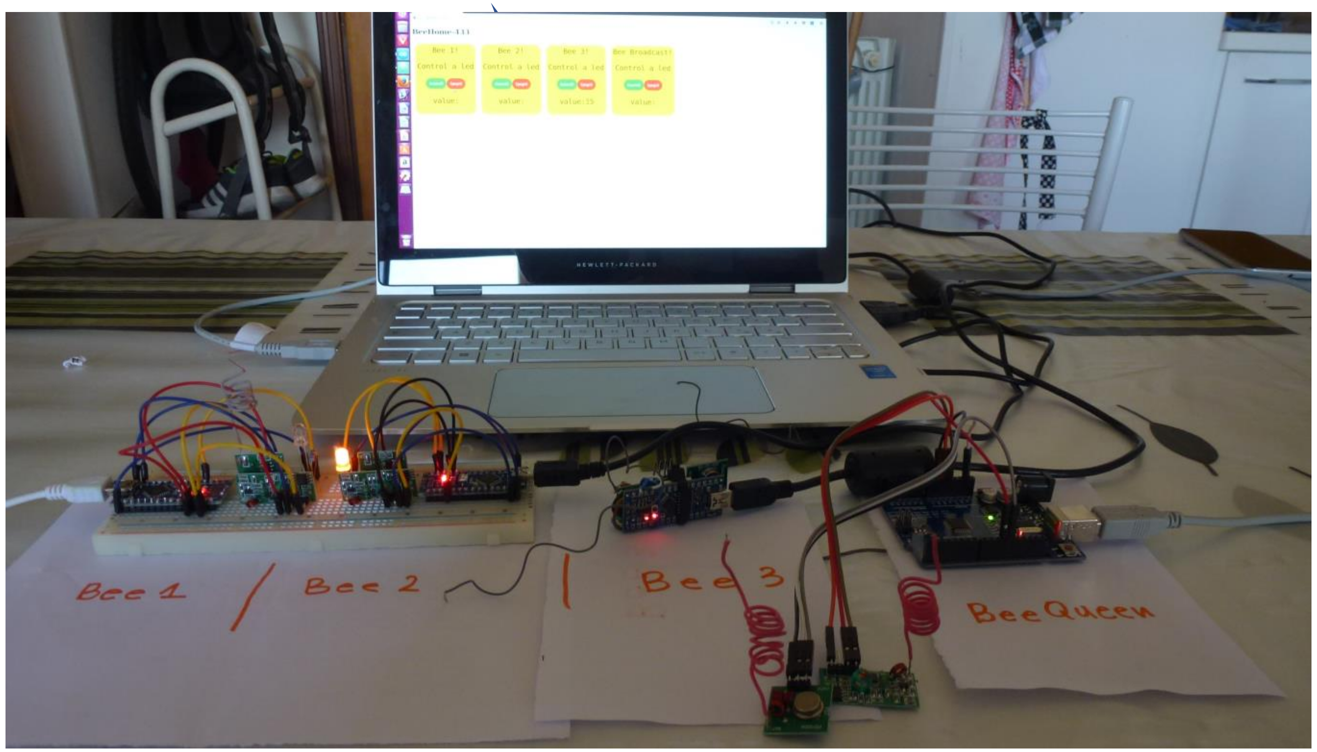
Improvements

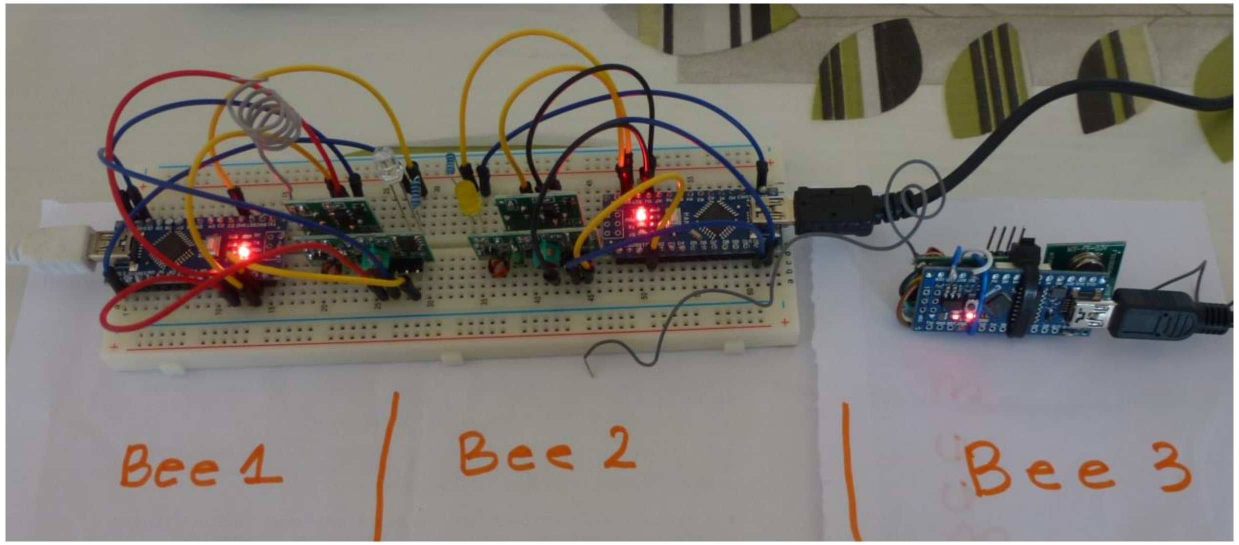
- Assignment dynamically the addresses to *Bees* (like DHCP) instead of static assignement.
- Add a “Media Access Control”.
- Adopt a Wireless communication from *BeeQueen* to *BeeHive instead of using the* serial port.
- Send commands to *BeeQueen* using a remote control (e.g. gate remote control)
- Add 12v power supply to 433Mhz transmitter for obtaining longer range communications.
- Send Telegram notifications to the users.

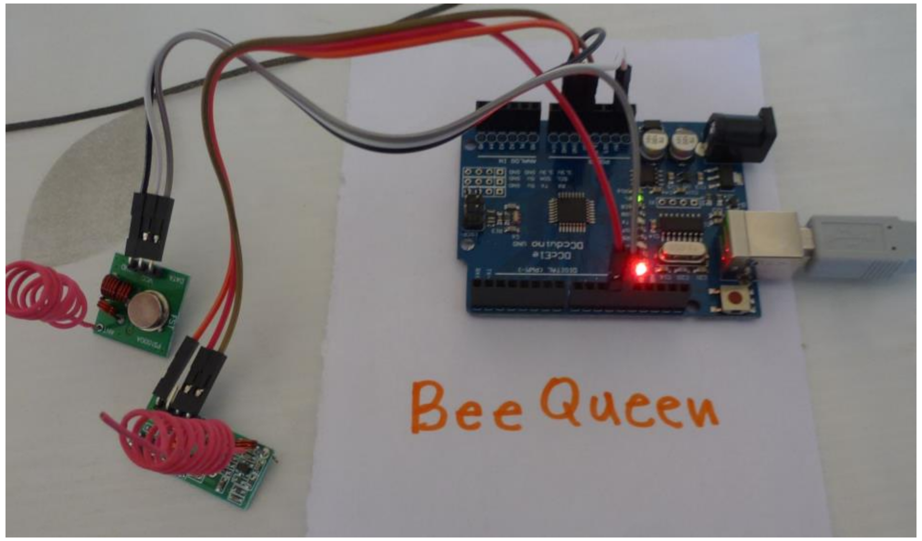
Demo: video on You Tube

https://youtu.be/yvfxI-R9Ju4v

Bee1 Bee 2 control a led
Bee 3 sends integer values







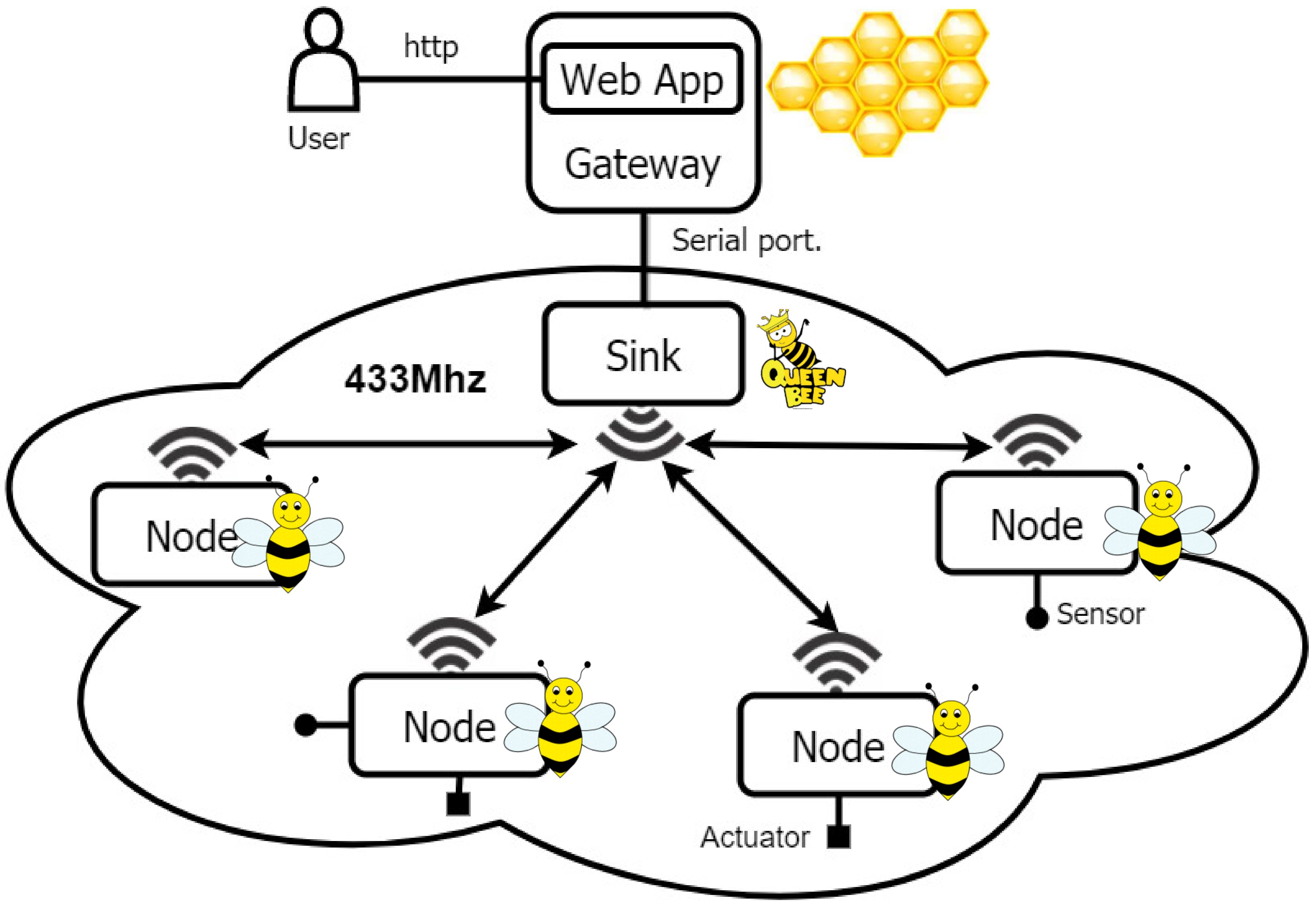
References

GitHub: https://github.com/dido18/beehouse-433

- [1] RadioHead library, http://www.airspayce.com/mikem/arduino/RadioHead/
- [2] 433Mhz antenna http://www.instructables.com/id/433-MHz-Coil-loaded-antenna/
- [3] VirtulaWire(deprecated), http://www.airspayce.com/mikem/arduino/VirtualWire/
- [4] RcSwitch, https://github.com/sui77/rc-switch
- [5] Other: https://andreasrohner.at/posts/Electronics/New-Arduino-library-for-433-Mhz-AM-Radio-Modules/

Tutorials

- http://randomnerdtutorials.com/rf-433mhz-transmitter-receiver-module-with-arduino/
- https://www.liwen.id.au/arduino-rf-codes/


User
http
Web App
Gateway
Serial port.
Sink
QUEEN BEE
433Mhz
Node
Node
Sensor
Node
Node
Actuator