May 2017

RHS Project

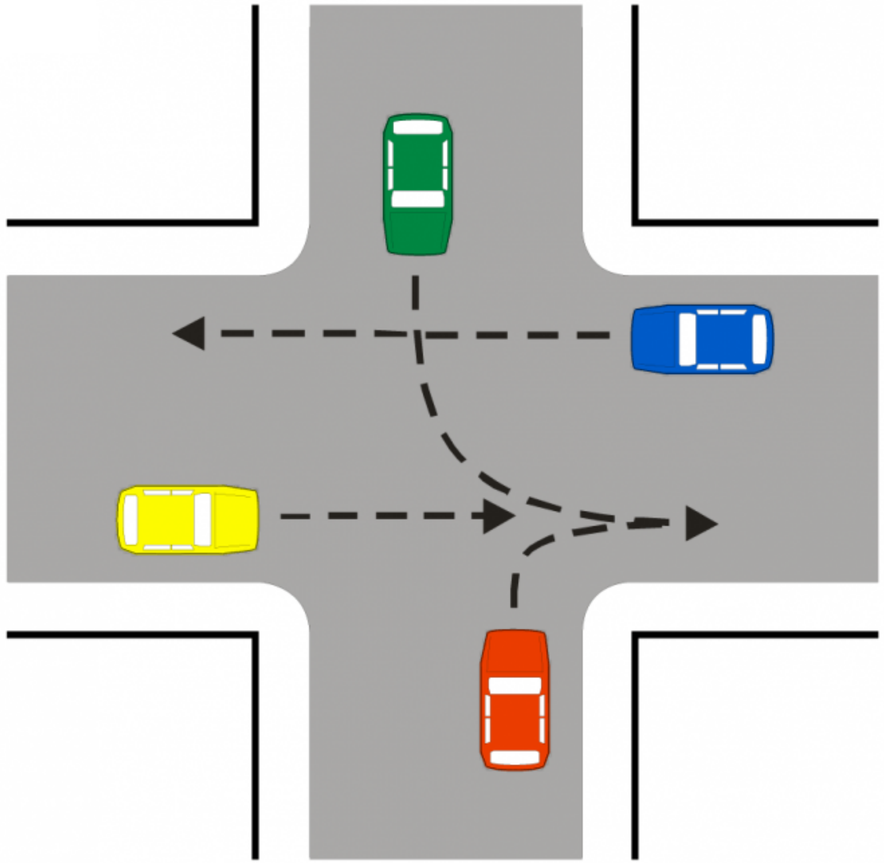
DRIPS

Decentralized Relative Inertial Positioning System

Daniele Di Sarli
daniele.disarli@gmail.com

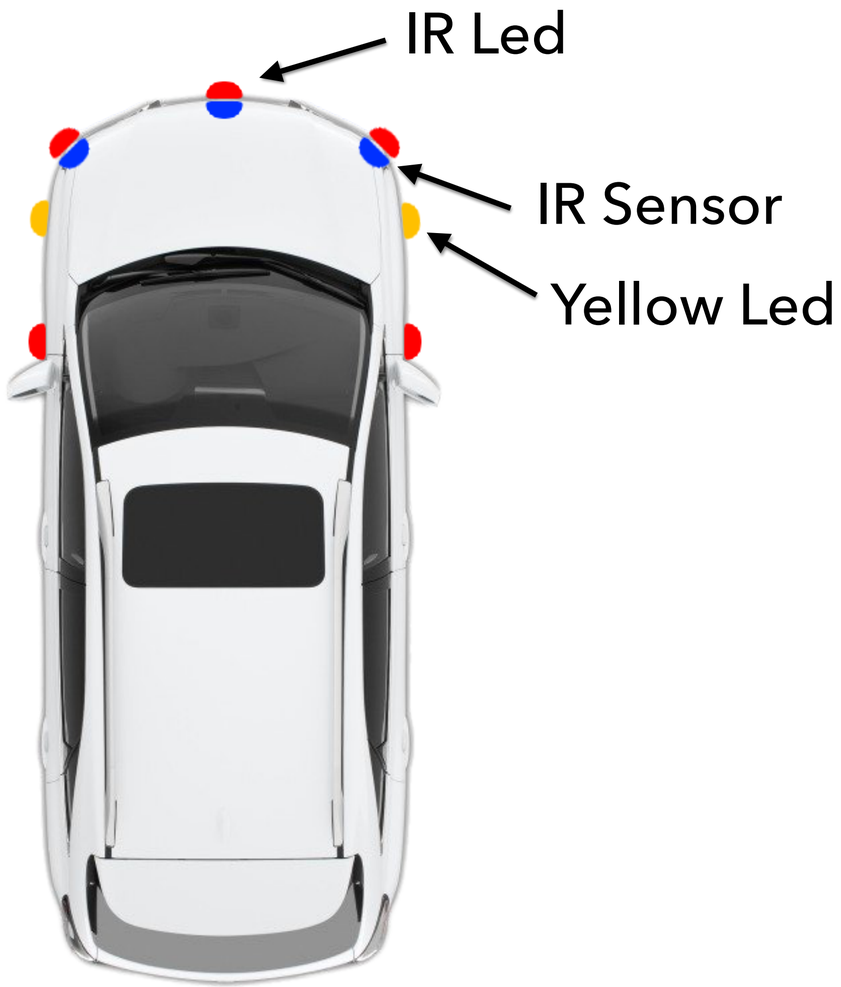
Diego Giorgini
diego.giorgini@icloud.com

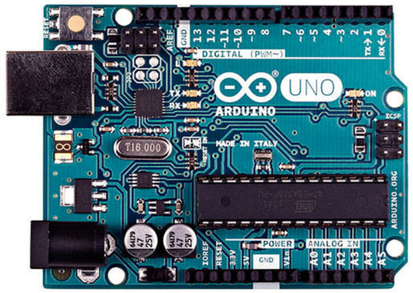
Goal



- To create a detection, communication and coordination system between small-scaled cars in a crossroad without any centralised infrastructure.
- Each car must be able to:
  - detect other vehicles around it
  - communicate with them in order to compute the precedence according to real road-rules, or to signal an emergency situation

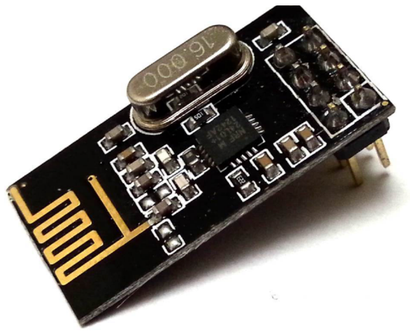
Car components





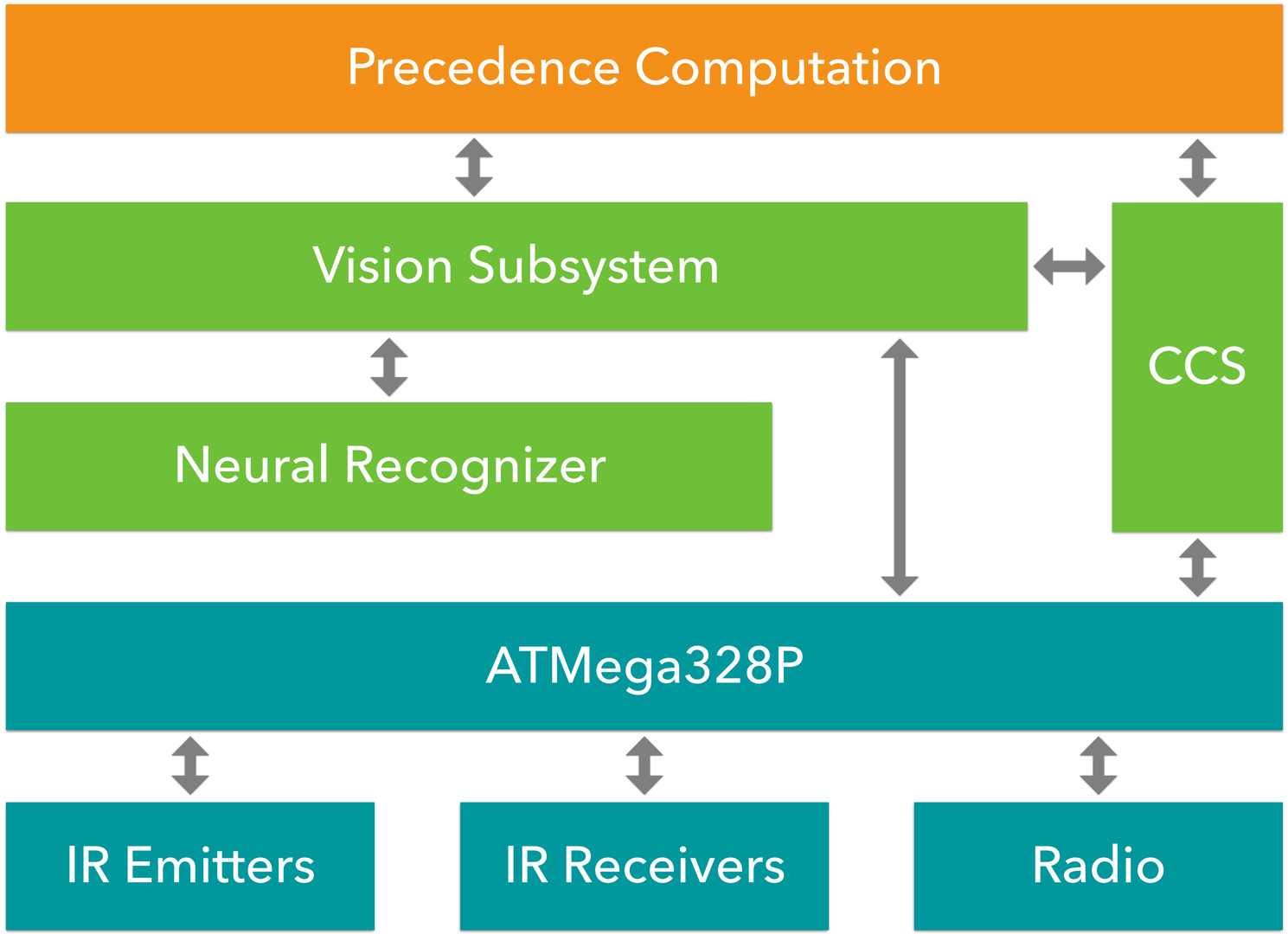
- Arduino Uno

- IR LEDs and Photodiodes

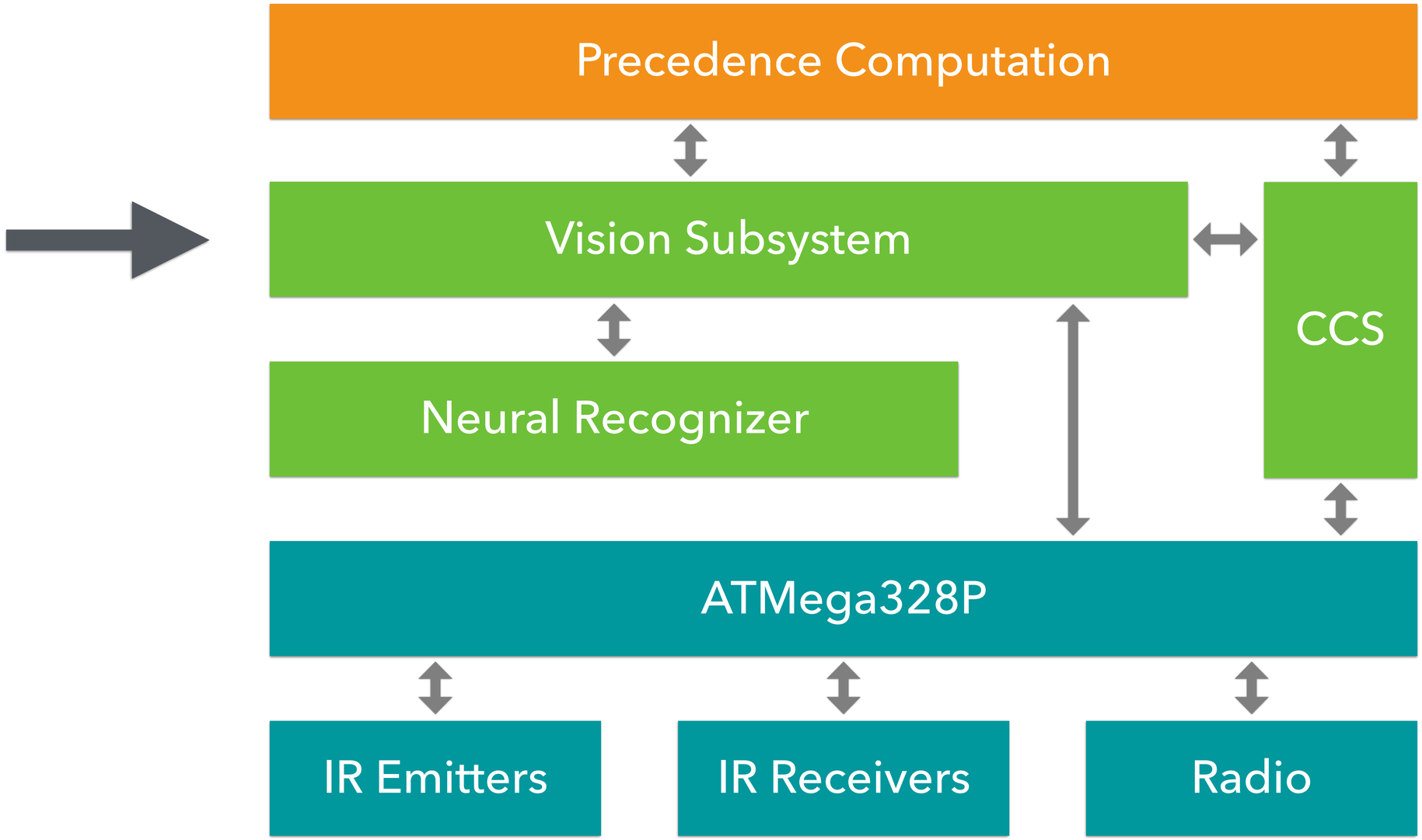


- NRF24L01+ Radio

System description

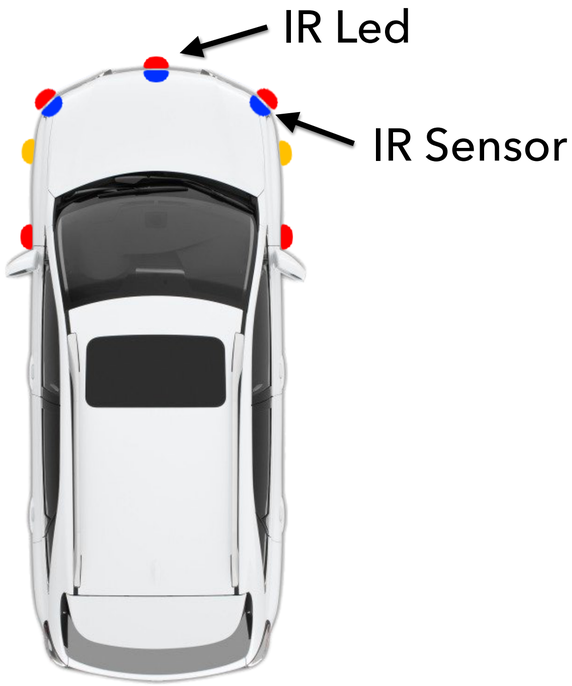


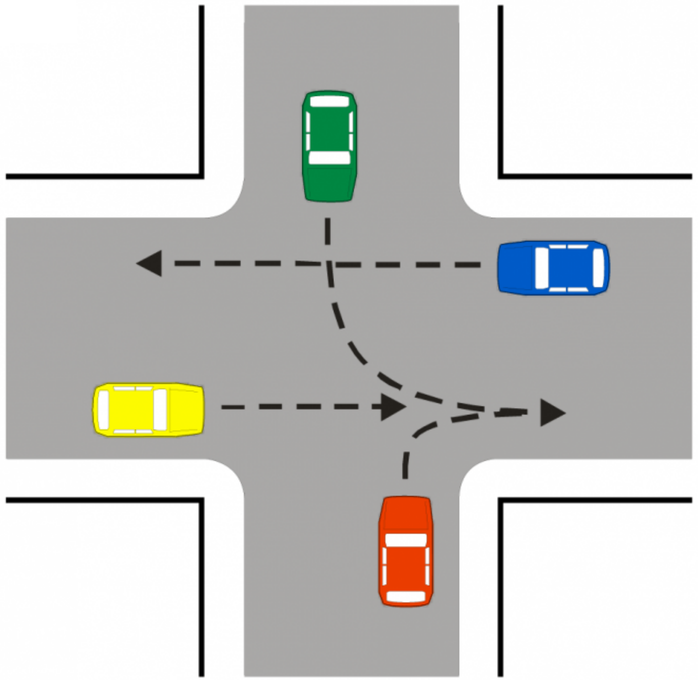
Vision Subsystem



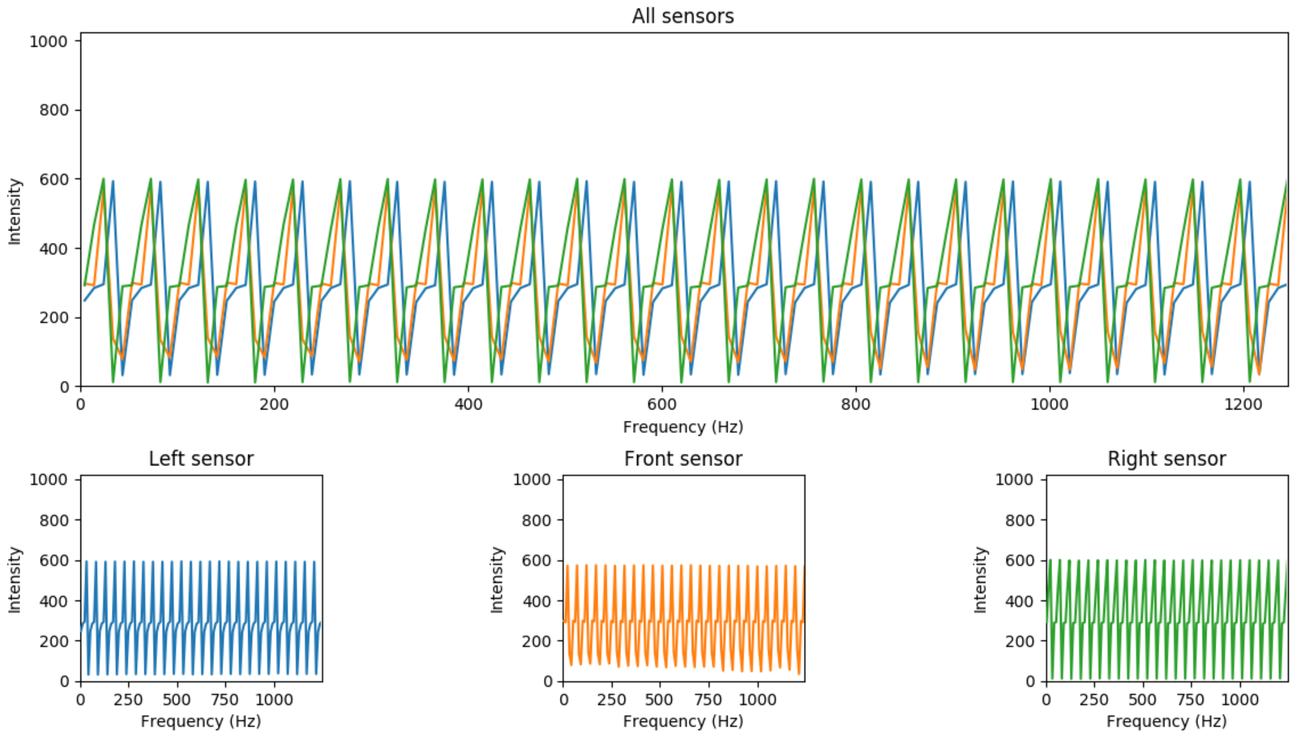
Vision Subsystem

How to detect other cars?



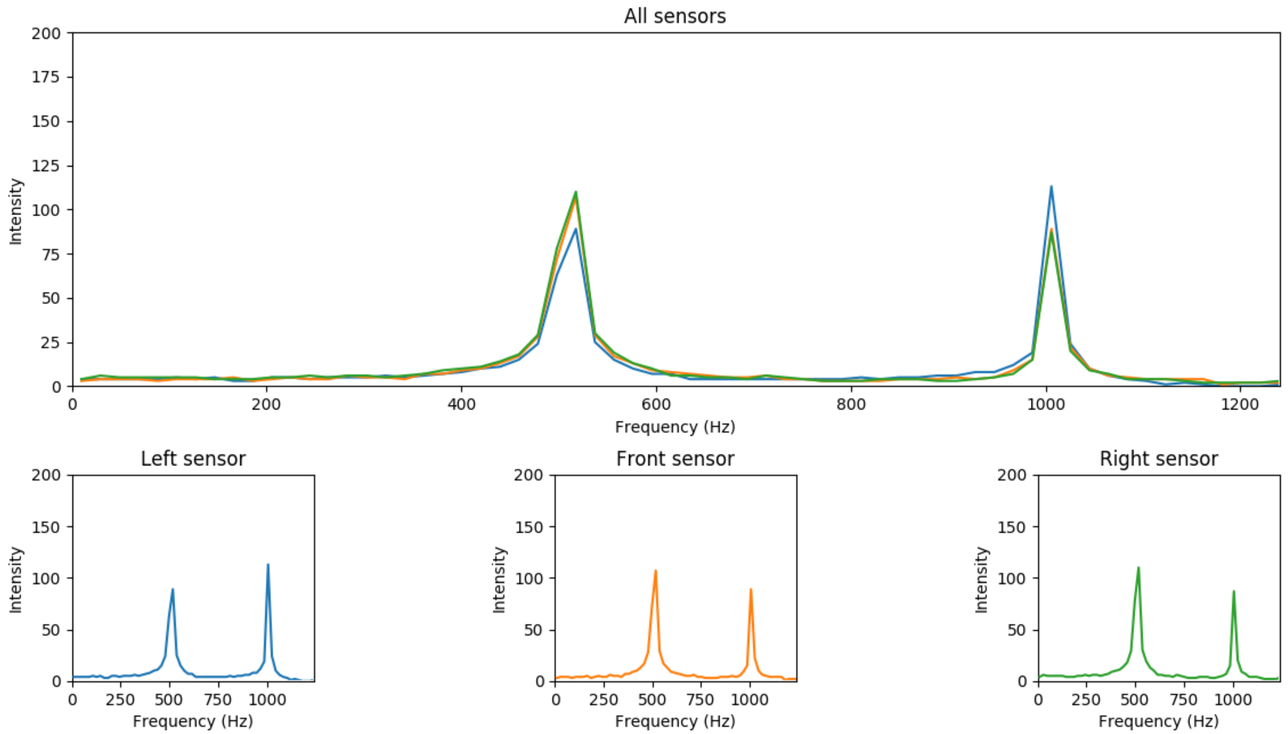


Sampling and FFT



1. Each LED blinks at a different frequency (60 Hz - 1 KHz)
2. We receive three independent directional signals
3. We apply the Fast Fourier Transform to the sampled data to extract the frequencies

Sampling and FFT



1. Each LED blinks at a different frequency (60 Hz - 1 KHz)
2. We receive three independent directional signals
3. We apply the Fast Fourier Transform to the sampled data to extract the frequencies

Implementation

Emission

- common hardware timer (T = 100 µs)
- interrupt handler (execution time must be much lower than T!)

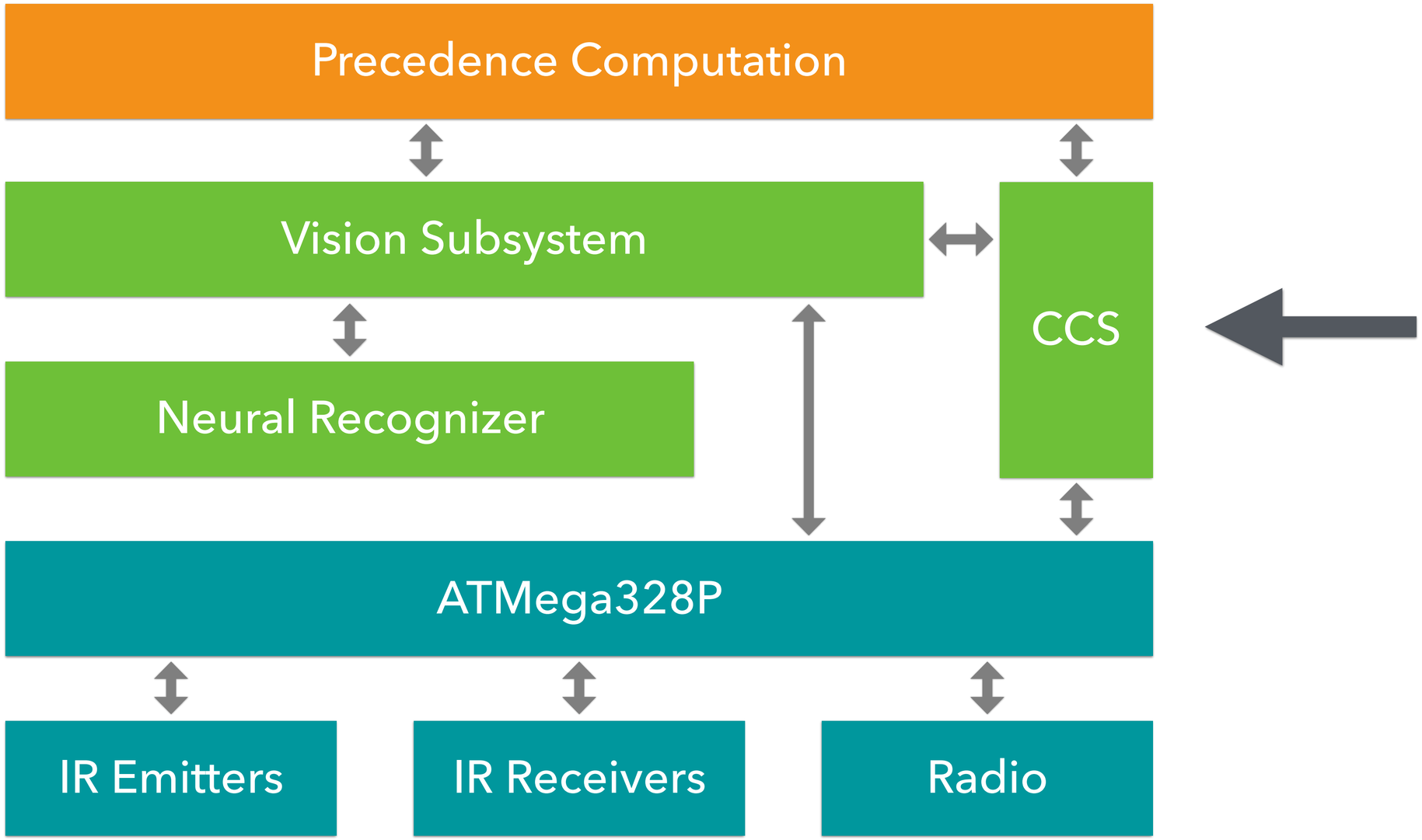
Sampling

- can't be done in the interrupt handler
- high precision obtained by synchronising with micros() instead of delayMicroseconds()

Fourier Transform

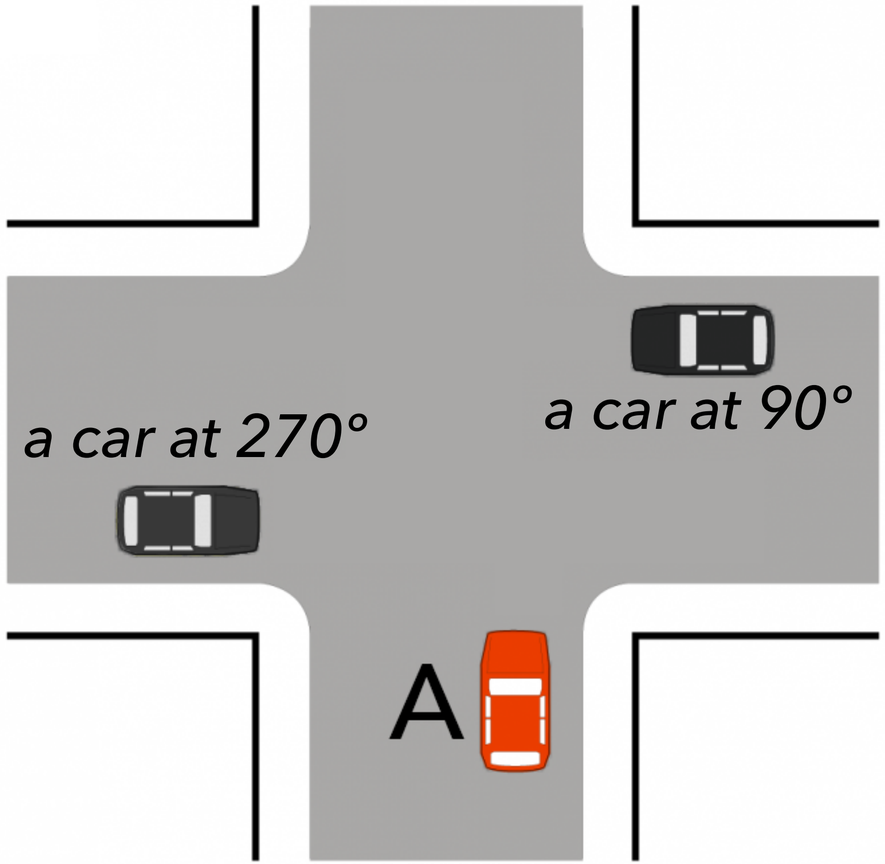
- fast FHT library written in assembly
- we extended the library with some methods to improve the result, like
  - fht_constant_detrend

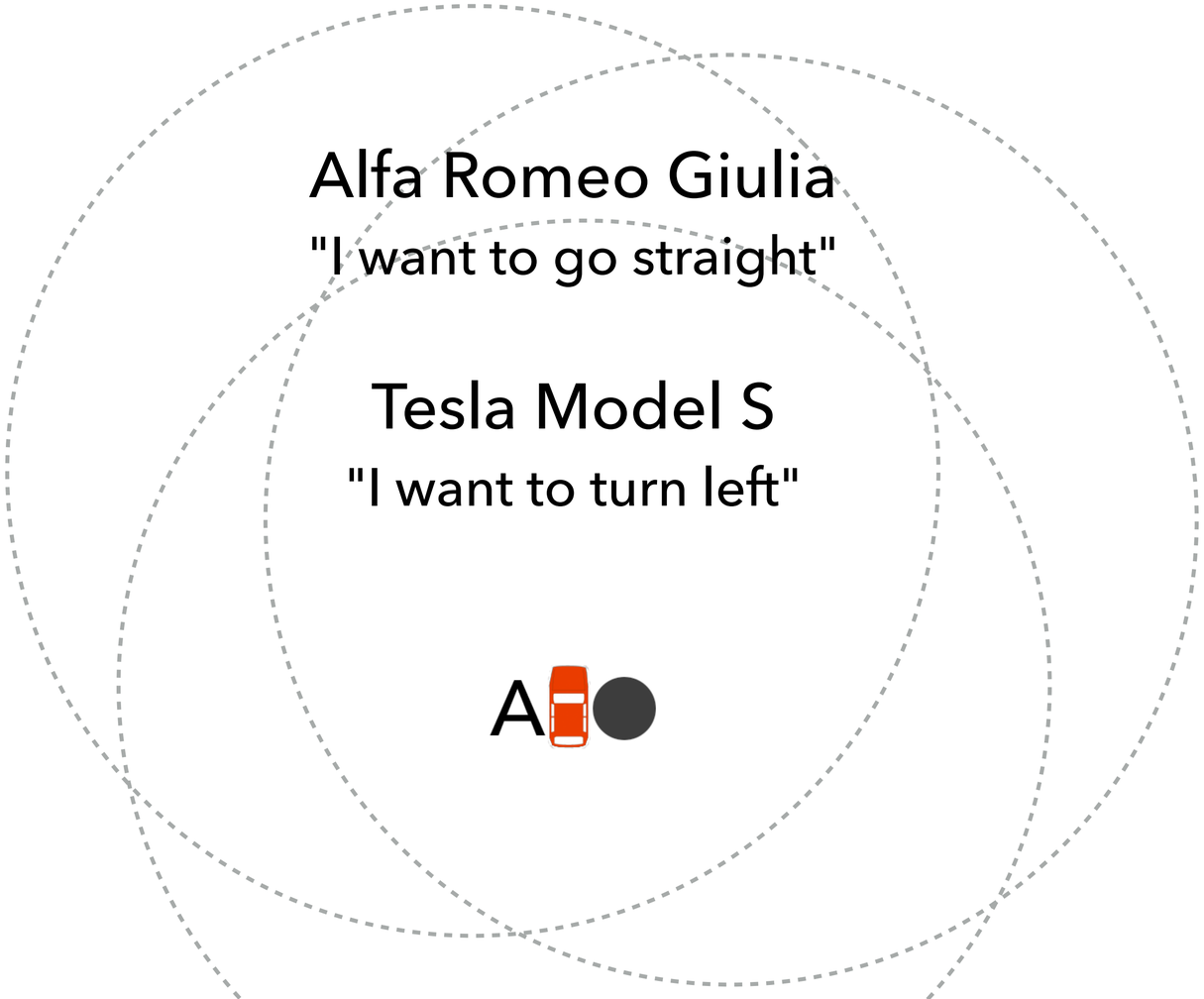
Car Communication System



Car Communication System

How to pair nodes of the network with the detected cars?





Overview

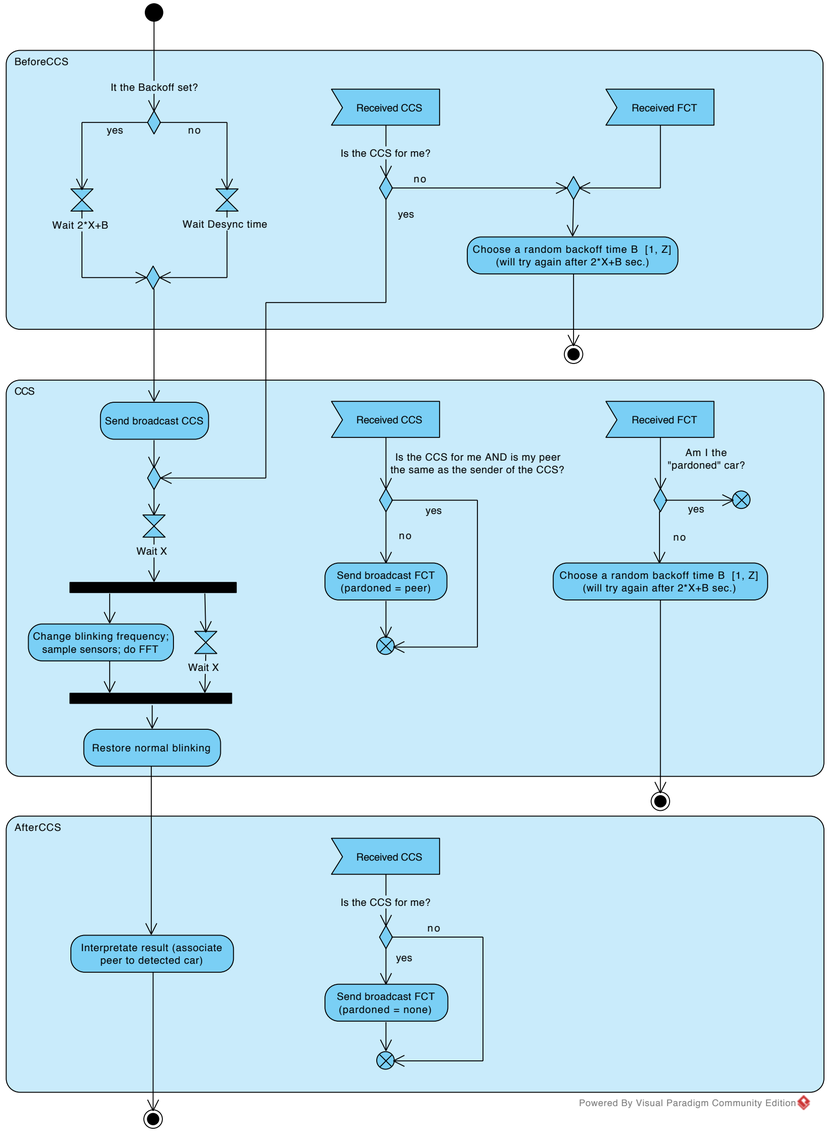
Functions:

- Make cars discoverable over the network while broadcasting their information
- Ask a node to blink at a special frequency in order to pair it to a detected car

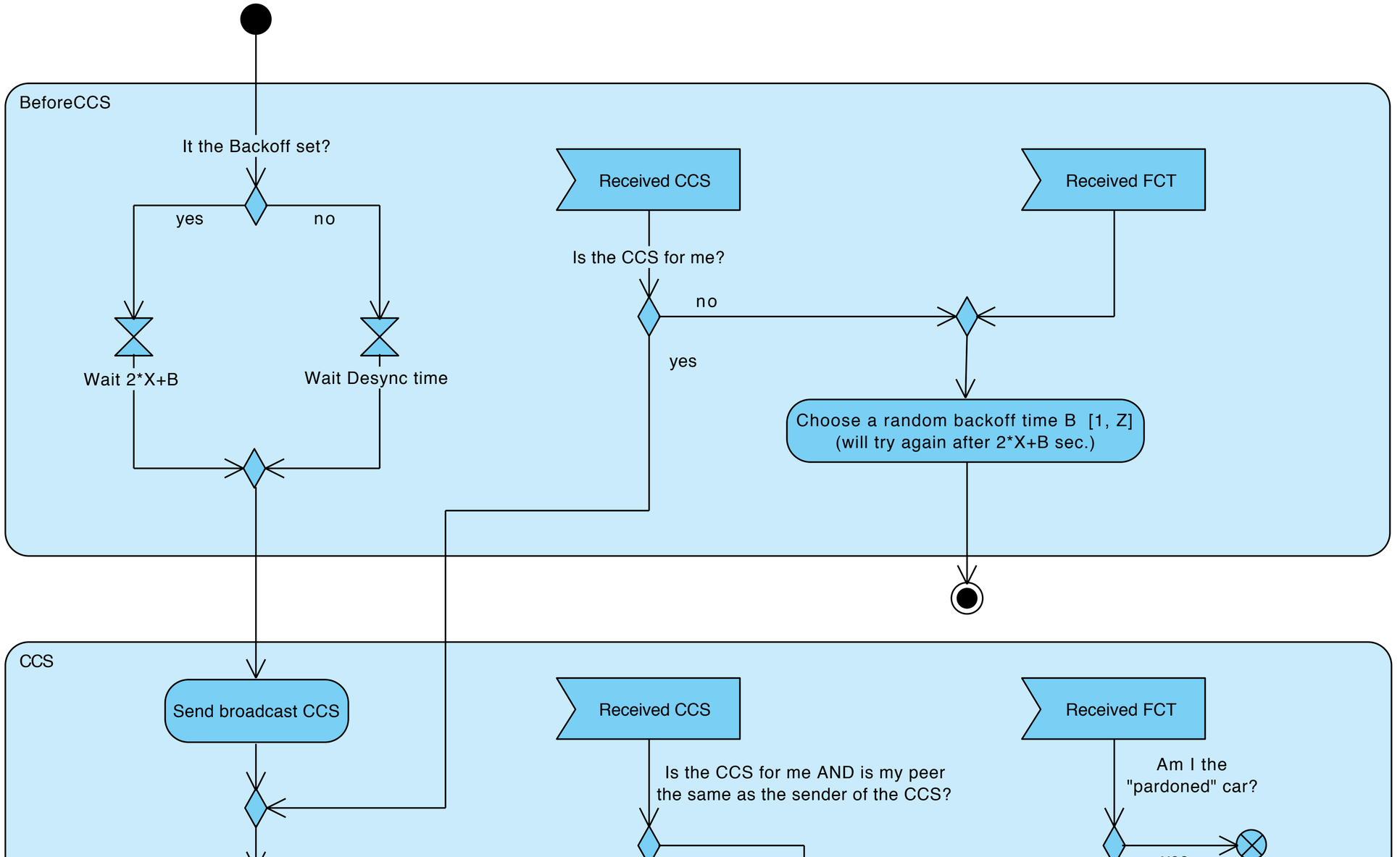
Messages:

- **KeepAlive**: sent to give the network the car's information
- **CCS**: to ask a node to start the pairing
- **FCT**: to halt other pairing procedures

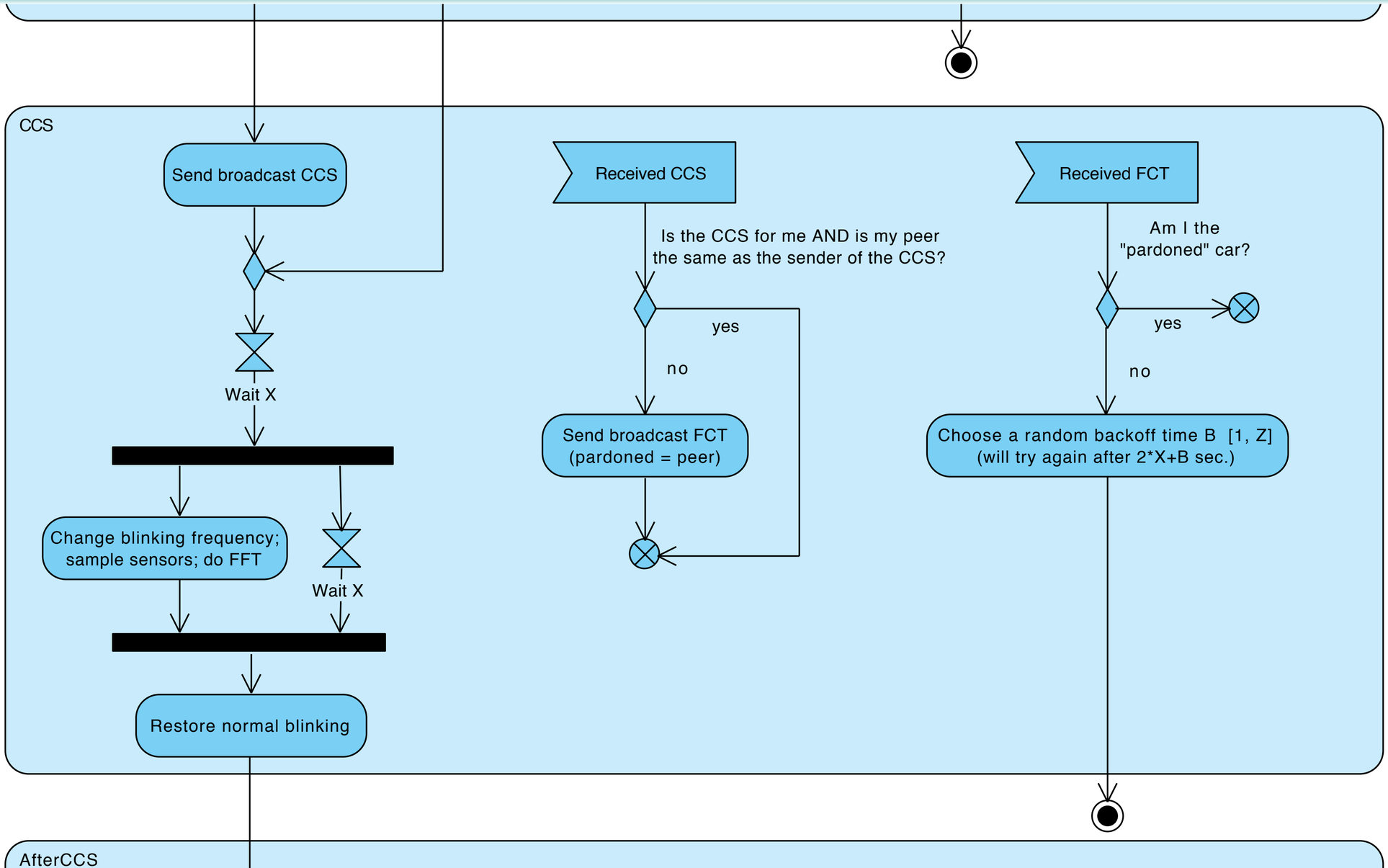
Activity Diagram



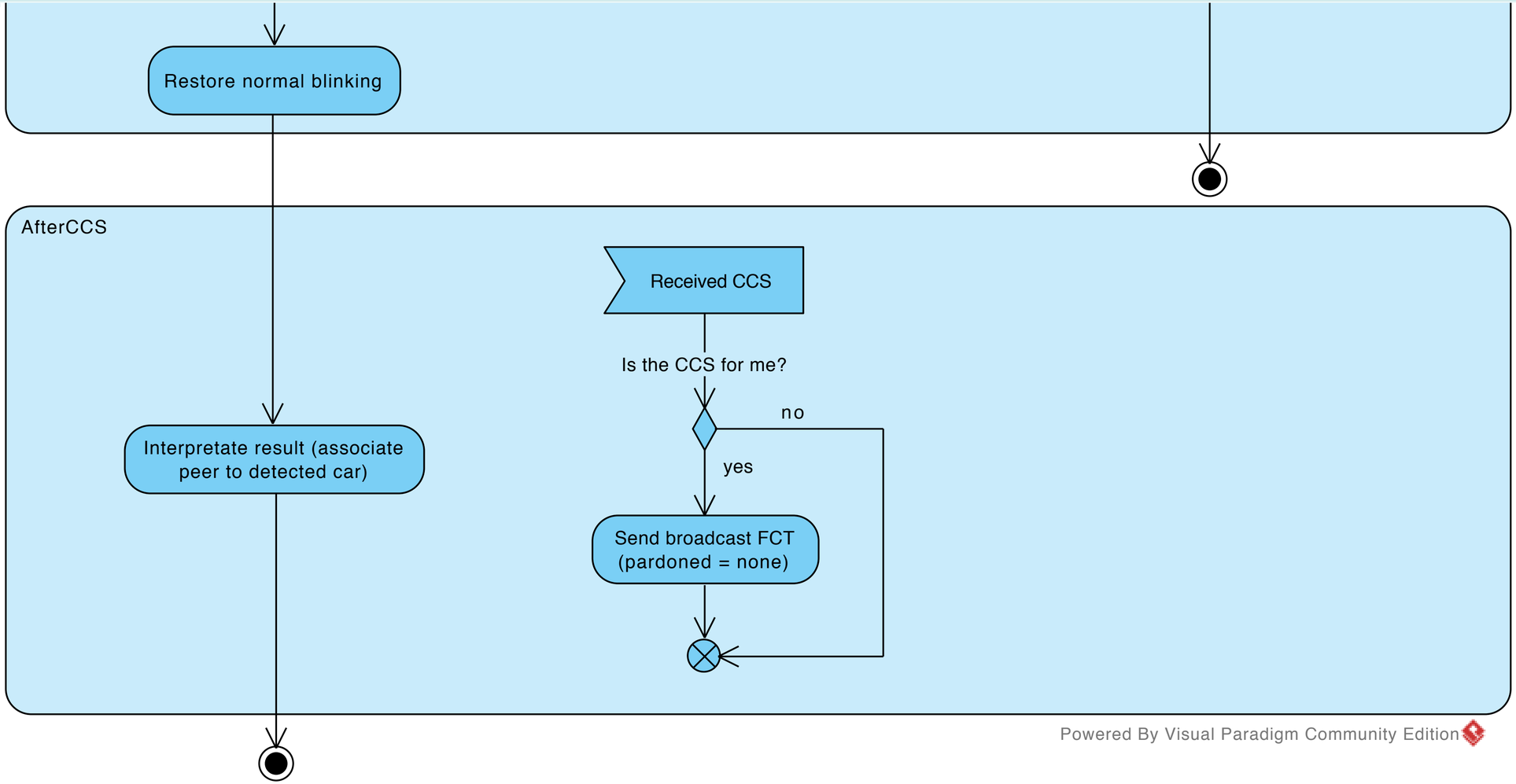
Activity Diagram



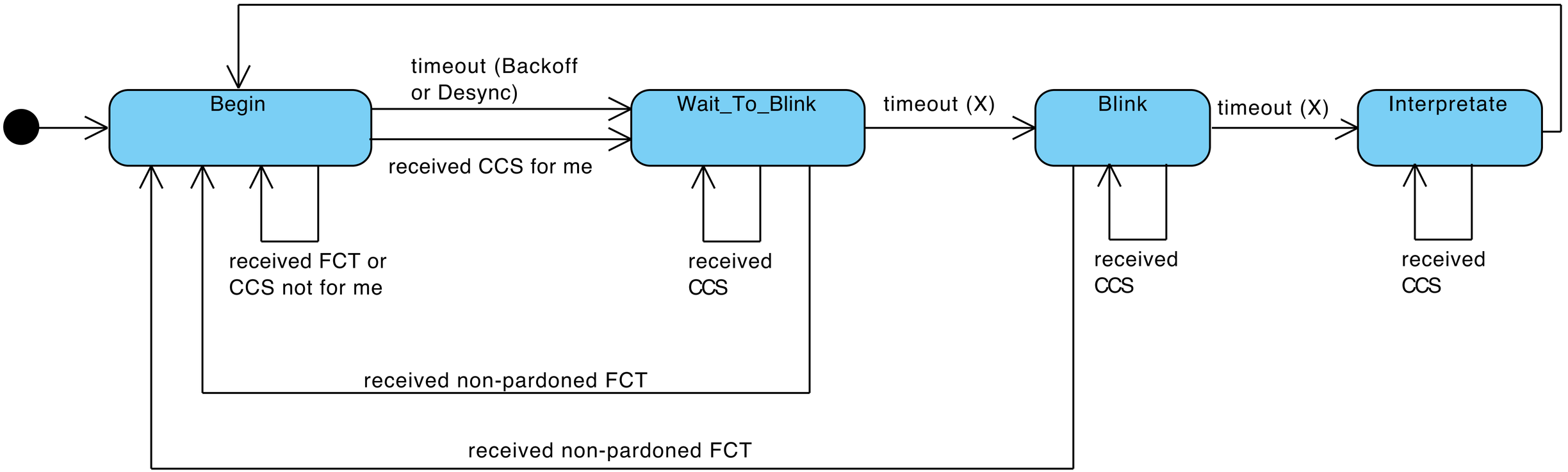
Activity Diagram



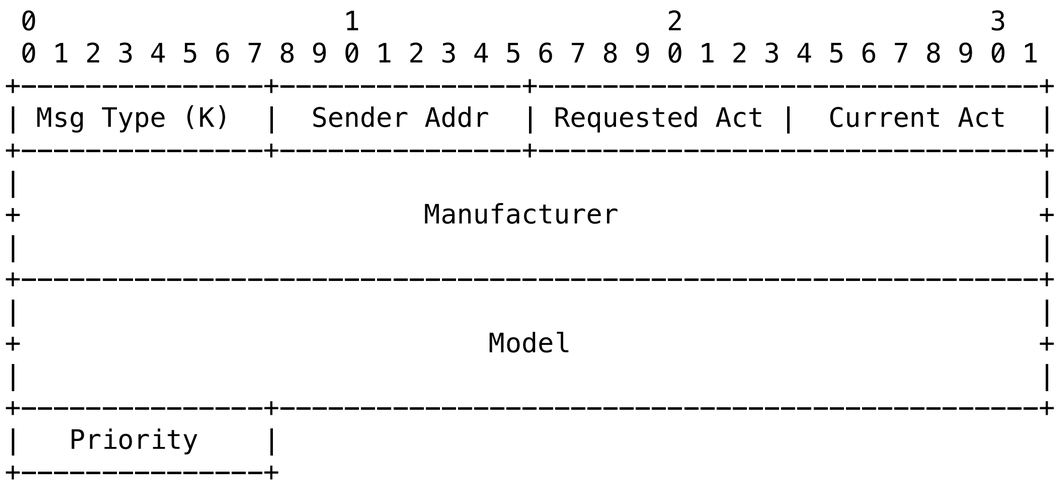
Activity Diagram



State Diagram



KeepAlive message



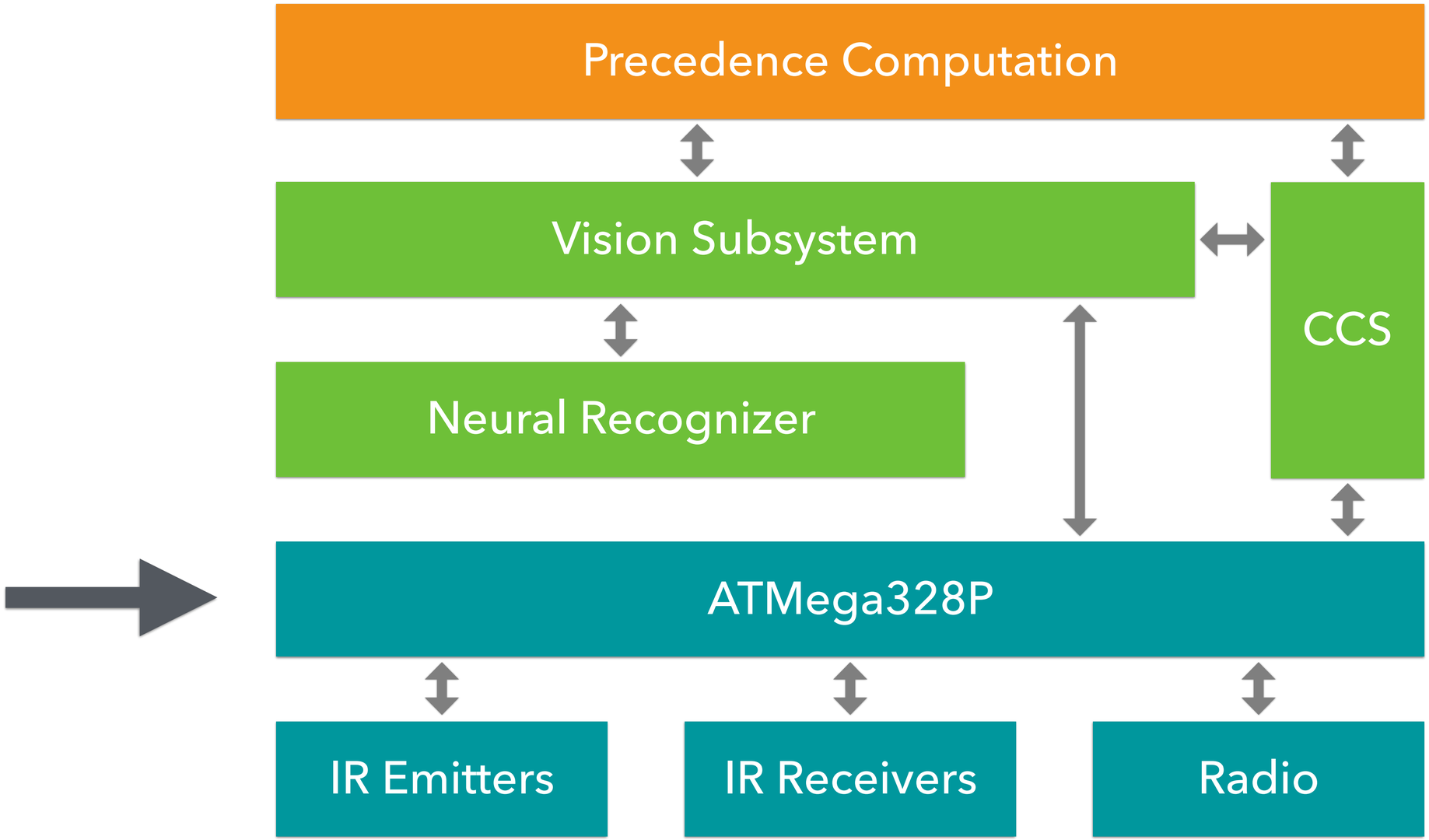
Three purposes:

- Discoverability
- Action broadcasting
- Information expiration

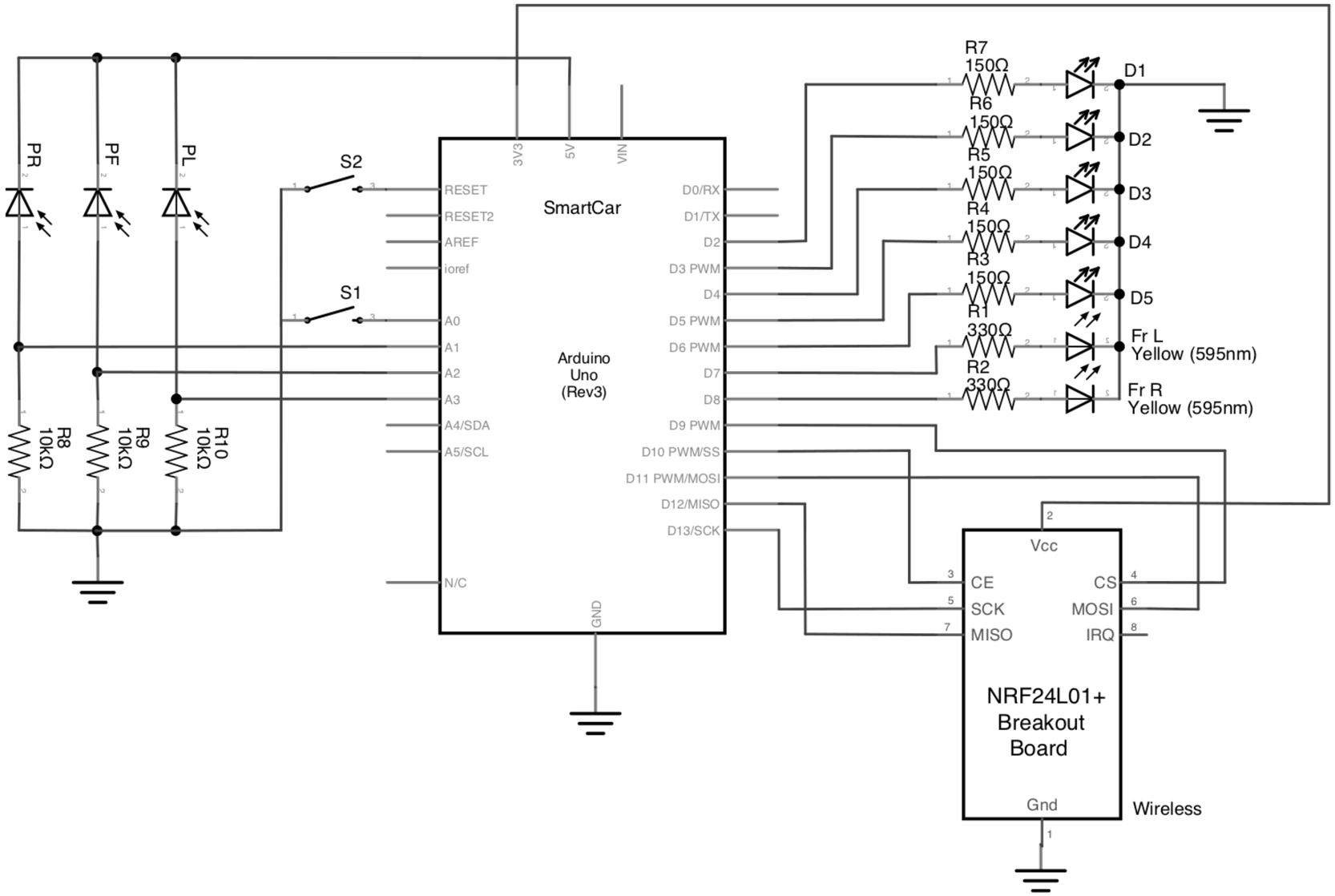
Specification

```
# RFC: CCS

## CAR COMMUNICATION SYSTEM
Specifiche del protocollo
Maggio 2017
Daniele Di Sarli, Diego Giorgini

## Funzionalità
Questo protocollo fornisce due funzionalità:

 - *ACQUISIZIONE:*
   conoscere le caratteristiche non rilevabili con il sottosistema visivo delle auto nelle
   vicinanze, come il modello dell'auto e le sue intenzioni;
 - *ASSOCIAZIONE:*
   associare un auto da cui si è ricevuto un messaggio sulla rete wireless con un'auto
   rilevata con il sottosistema visivo.

## Notazione
Di seguito utilizziamo nomi di parametri che, per brevità, non coincidono con i nomi utilizzati
nell'implementazione di riferimento del protocollo. Si elencano qui le associazioni tra i nomi
utilizzati in questo documento e il loro corrispettivo dell'implementazione.

 * X: TIMESPAN_X
 * Z: TIMESPAN_MAX_BACKOFF

## Descrizione del protocollo

I messaggi inviati sono di tre tipi:

 * KeepAlive:
     inviati periodicamente per lo scoperta e il mantenimento dei nodi nella rete
 * CCS (Car Communication Signal):
     per avviare la procedura
 * FCT (Force Communication Termination):
     per notificare altre auto che un'altra procedura è in corso nella rete

Supponiamo che l'auto A abbia rilevato tramite il sottosistema visivo tutte le auto nell'incrocio.
Il passo successivo è mettere in relazione ogni auto rilevata con i dati acquisiti sulla rete wireless.

Per fare ciò, l'auto A deve iniziare la procedura di CCS.

 1. L'auto invia in broadcast un pacchetto CCS con specificato l'indirizzo dell'auto
    che A vuole associare.
 2. L'auto attende X secondi.
 3. L'auto entra nella fase di rilevamento della frequenza tramite sensori IR.
    L'auto si aspetta di osservare per una durata di altri X secondi una frequenza
    di K Hz su uno o più sensori. Contemporaneamente, inizia anch'essa a far
    lampeggiare i led con una frequenza di K Hz per una durata di X secondi.
 4. Dopo lo scadere degli X secondi, l'auto interpreta i dati rilevati e li associa
    al nodo a cui è stata fatta la richiesta. Dopodiché torna nello stato di
    operatività standard.

In ogni momento della procedura di CCS, l'auto A rimane in ascolto di messaggi FCT. In
caso di ricezione di tale messaggio, e in caso in cui il Pardoned Address del messaggio
sia diverso da quello dell'auto A, la procedura viene interrotta e ripresa dopo un
tempo di 2X sommato a un backoff casuale compreso tra 1 e Z millisecondi.

Ogni altra auto nell'incrocio, alla ricezione del CCS di A:

 1. Determina se attualmente è impegnata in una procedura di CCS con un qualsiasi peer.
     a. Se sì e se quel peer non è A, oppure se sì e se il CCS non era per questa auto,
        allora risponde in broadcast con un FCT con specificato l'indirizzo del peer con
        cui è in corso la procedura già avviata.
     b. Se no, determina se il CCS era destinato a lei.
          i. Se no, termina
         ii. Se sì, attende X secondi da quando ha rilevato il CCS, quindi entra
             nella fase di rilevamento della frequenza tramite sensori IR.
             L'auto si aspetta di osservare per una durata di altri X secondi una frequenza
             di K Hz su uno o più sensori. Contemporaneamente, inizia anch'essa a far
             lampeggiare i led con una frequenza di K Hz per una durata di X secondi.

Durante tutto il tempo in cui si trova nello stato 1.b.ii., l'auto rimane in ascolto di
messaggi FCT. In caso di ricezione di tale messaggio, e in caso in cui il Pardoned Address
del messaggio sia diverso da quello dell'auto corrente, la procedura viene interrotta.

## Formato dei pacchetti
### KeepAlive

  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +---------------+---------------+---------------------------------+
 | Msg Type (K)  |  Sender Addr  | Requested Act |  Current Act  |
 +---------------+---------------+---------------------------------+
 |                                                                 |
 +                         Manufacturer                            +
 |                                                                 |
 +-----------------------------------------------------------------+
 |                                                                 |
 +                            Model                                +
 |                                                                 |
 +---------------+-------------------------------------------------+
 |   Priority    |
 +---------------+

 Msg Type      = 75 (carattere K)
 Sender Addr   = l'indirizzo che identifica l'auto che ha emesso il CCS.
 Requested Act = l'azione che l'auto vorrebbe fare
 Current Act   = l'azione che l'auto sta facendo a seguito della coordinazione con la rete
 Manufacturer  = il produttore dell'auto
 Model         = il modello dell'auto
 Priority      = (boolean) se l'azione richiesta dall'auto ha priorità su quelle delle altre auto

### CCS

  0                   1                   2
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
 +---------------+---------------+---------------+
 | Msg Type (C)  | Receiver Addr | Sender Addr   |
 +---------------+---------------+---------------+

 Msg Type = 67 (carattere C)
 Receiver Addr = l'indirizzo che identifica l'auto è cui è rivolto il CCS.
 Sender Addr = l'indirizzo che identifica l'auto che ha emesso il CCS.

### FCT

  0                   1
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
 +---------------+---------------+
 | Msg Type (S)  | Pardoned Addr |
 +---------------+---------------+

 Msg Type      = 83 (carattere S)
 Pardoned Addr = l'indirizzo che identifica l'auto che è attualmente impegnata nell'esecuzione
                 di una procedura CCS, e che quindi deve ignorare questo messaggio.
```

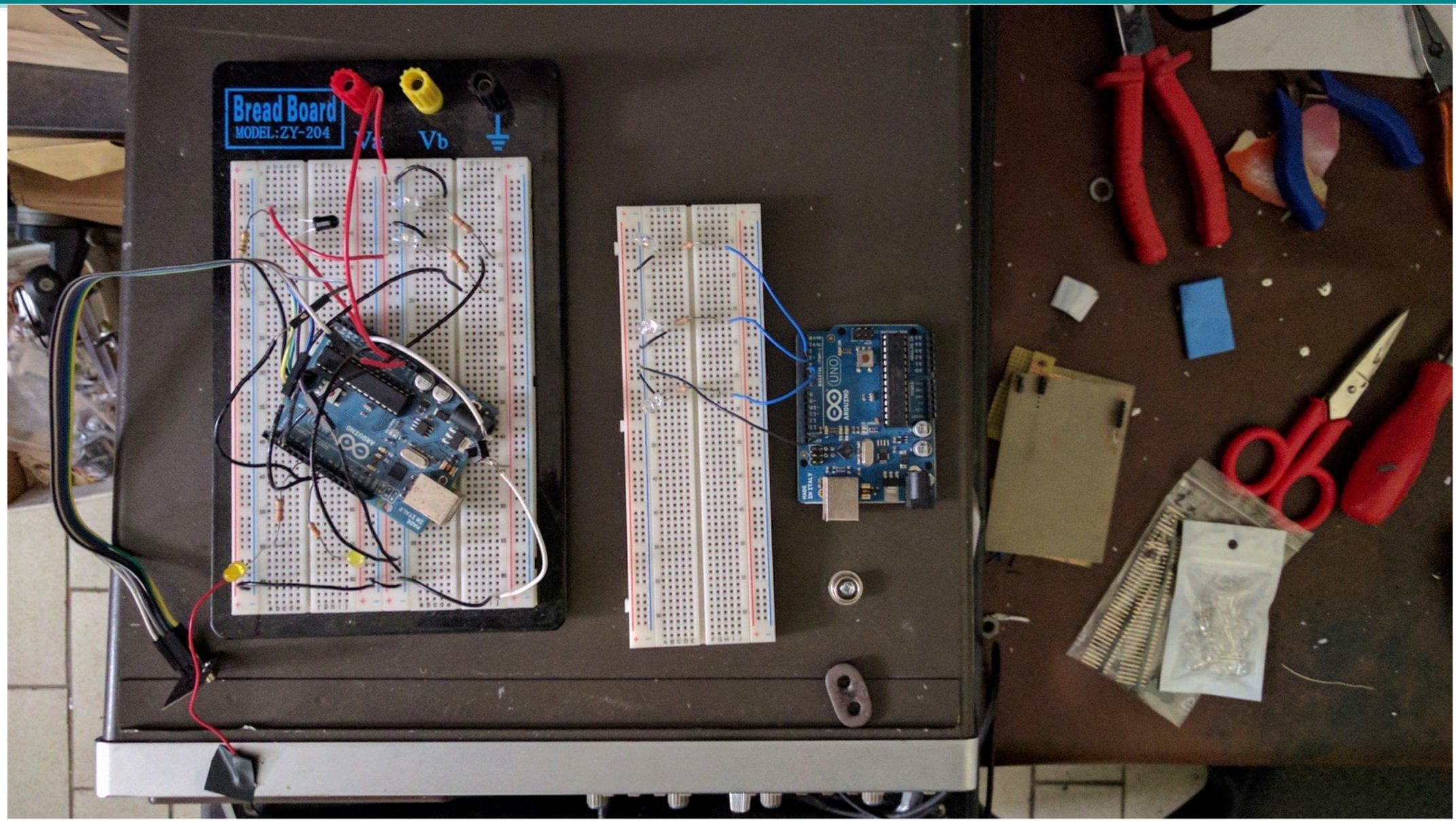
Hardware


Precedence Computation
Vision Subsystem
CCS
Neural Recognizer
ATMega328P
IR Emitters
IR Receivers
Radio

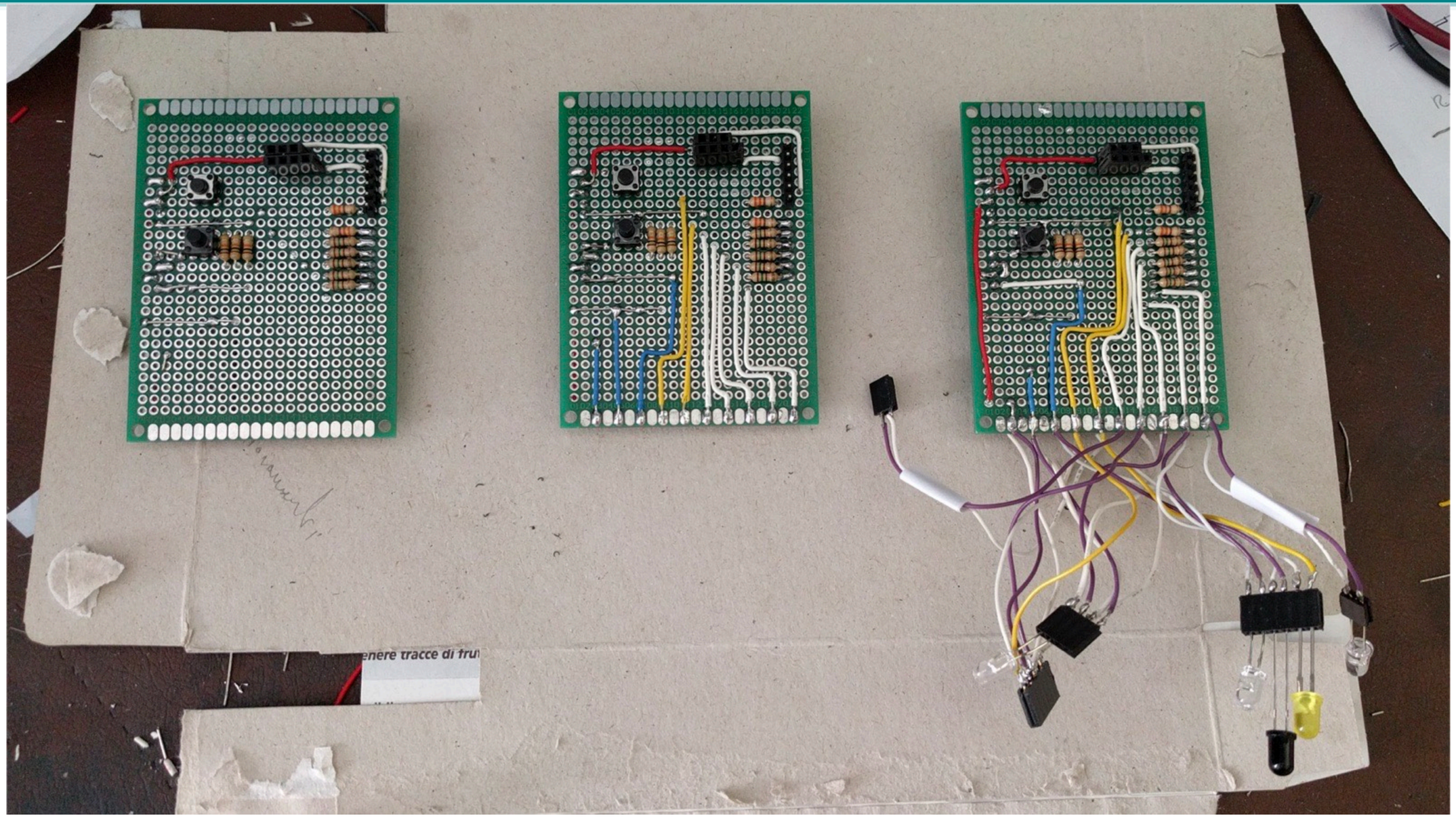
Circuit diagram



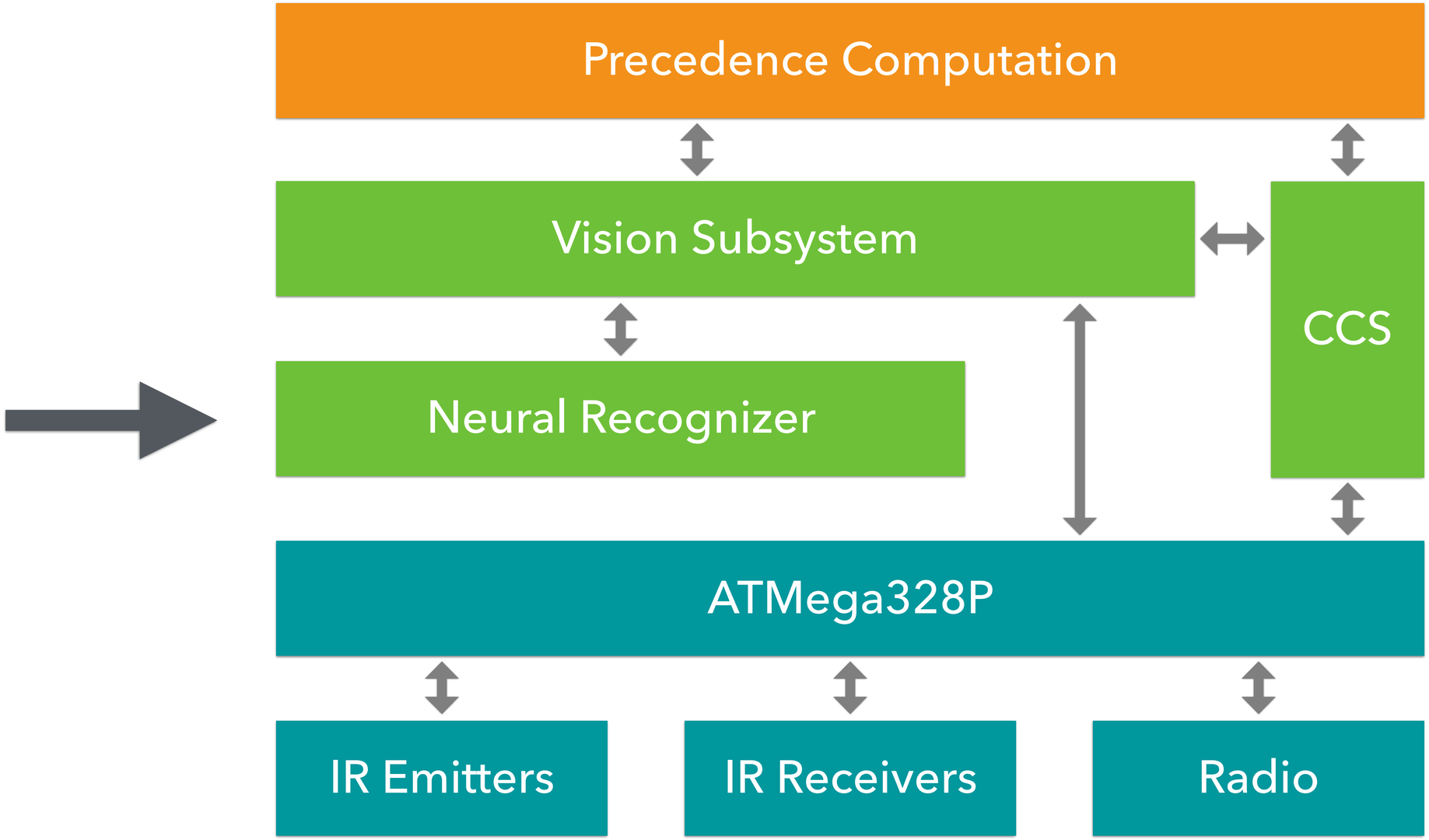
Prototype



Final boards

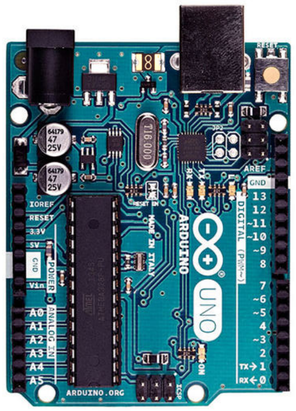


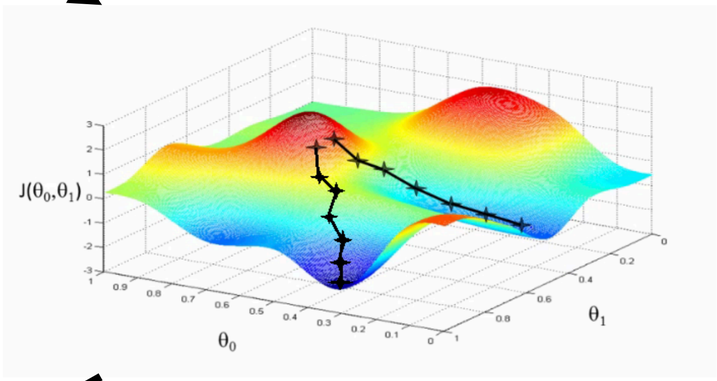
Neural Recognizer



Neural Recognizer

1. Observed frequencies

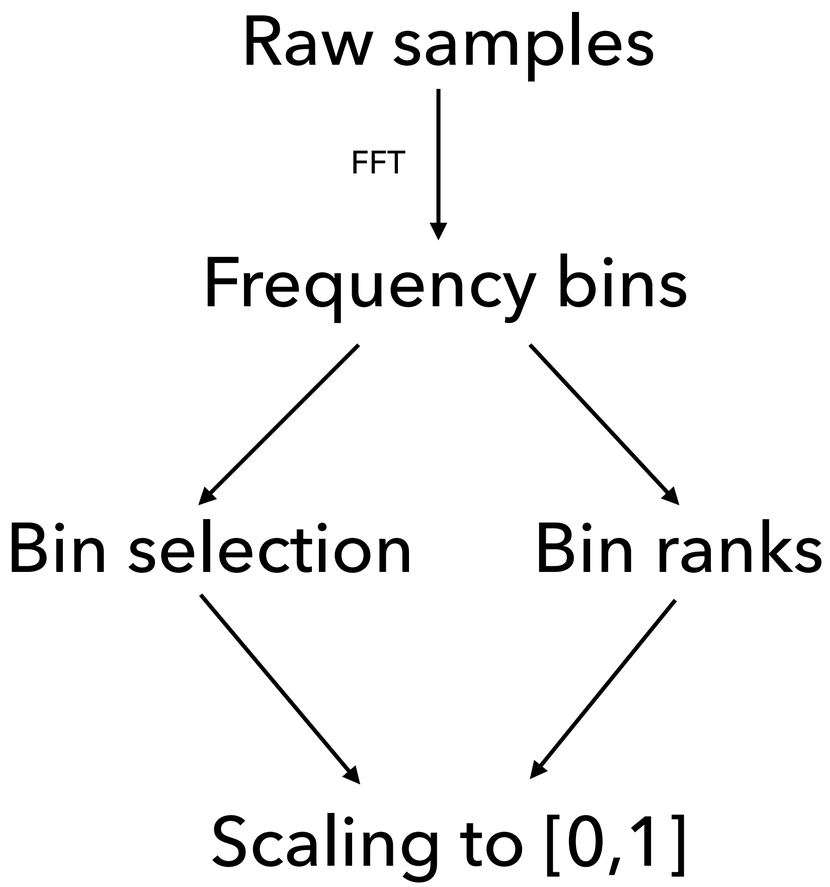


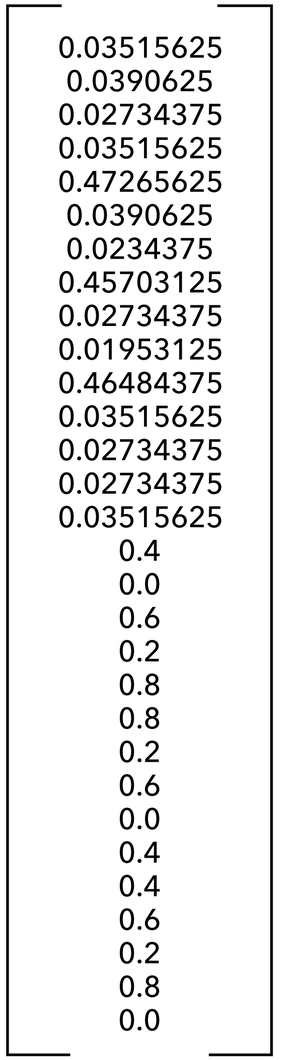


Training

2. Weights and Biases

Network input





Bins sensor 1

Bins sensor 2

Bins sensor 3

Ranks bin 1

Ranks bin 2

Ranks bin 3

Network output

[0, 0, 1, 0, 0, 0, 0, 0]

000 001 010 011 100 101 110 111

I see one car in front of me

I see a car on my left and a car in front of me

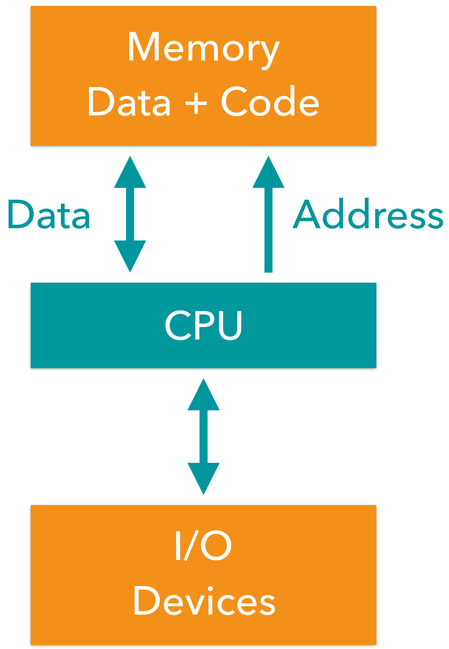
Memory considerations

- **Input**: a floating point vector of size 30 ⇒ 240 bytes
- **Weights**: a 30×8 floating point matrix ⇒ 960 bytes
- **Biases**: a floating point vector of size 8 ⇒ 32 bytes
- **Total**: 240 + 960 + 32 = **1.232 KB**

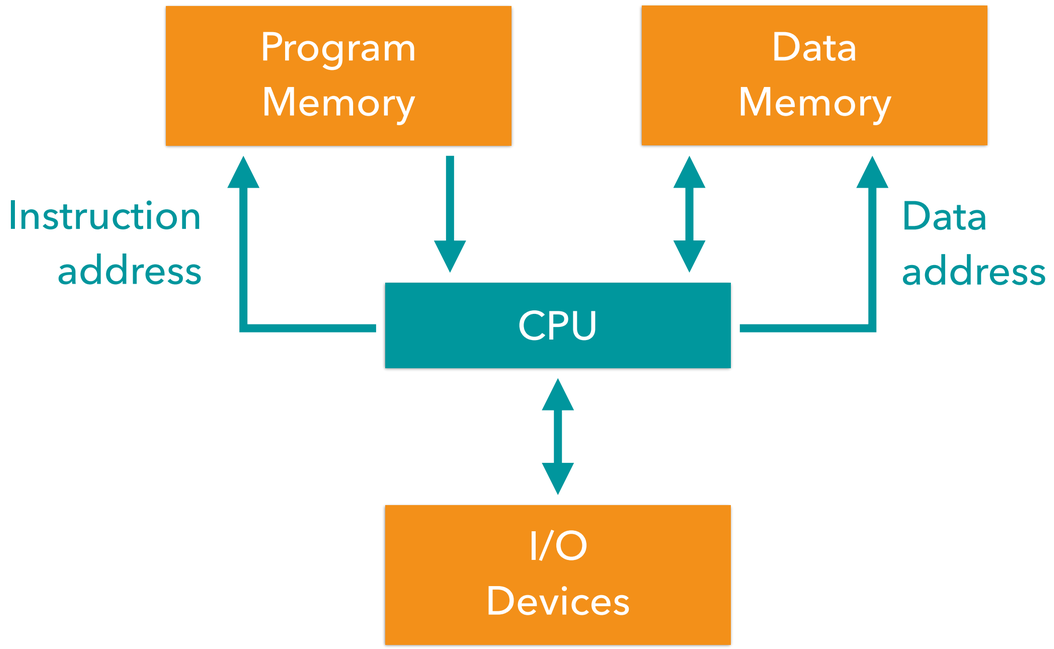
... more than half the total RAM!

Memory considerations

Solution: store the weights in Program Memory



Von Neumann Machine



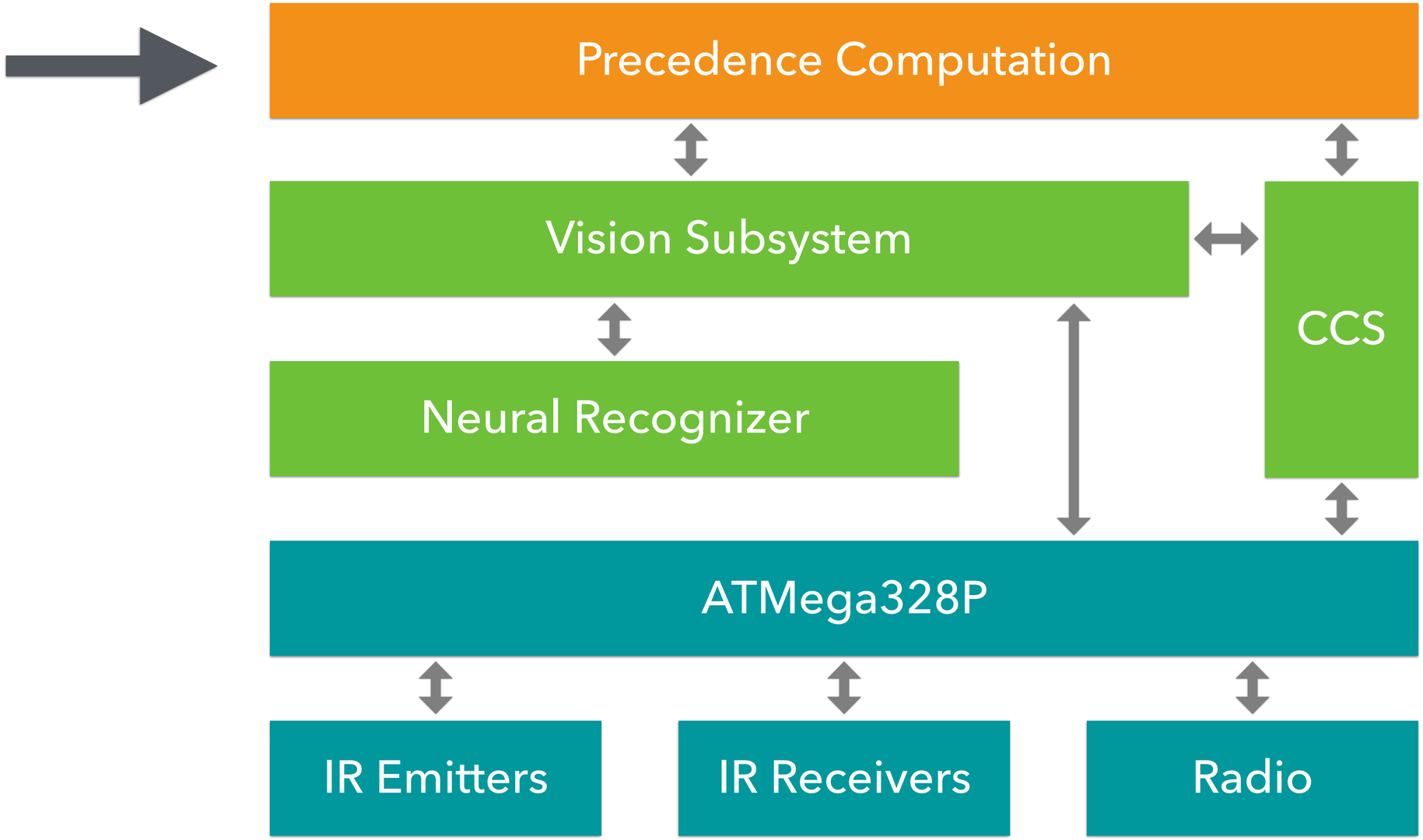
Harvard Machine

Results

- Training dataset size: 14.000
- Validation dataset size: 7000

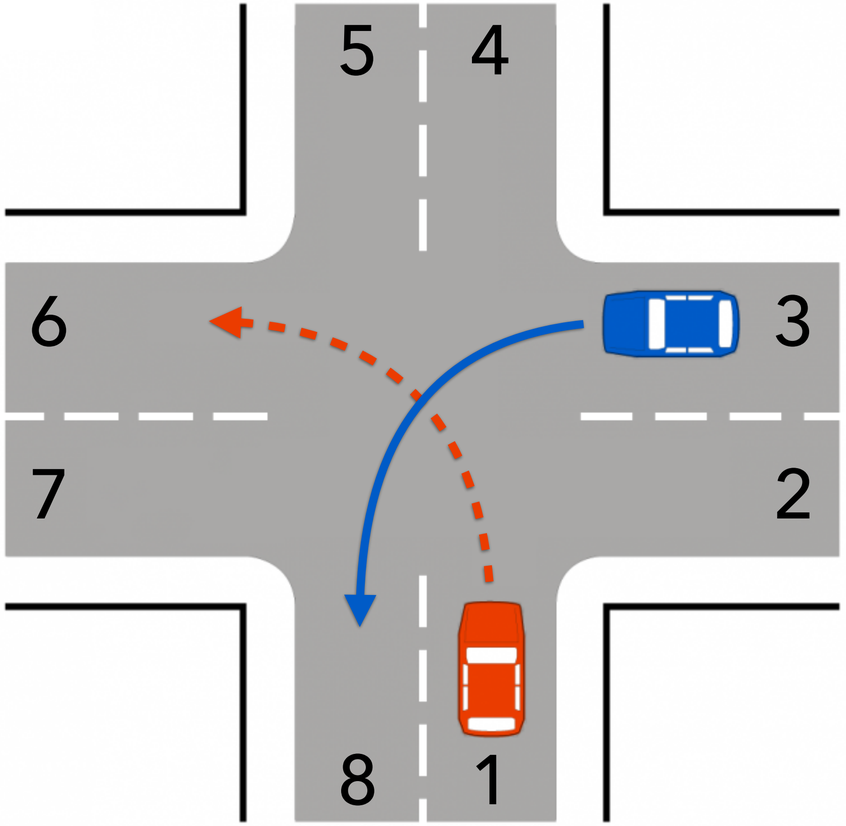
Accuracy: 0.988525

Precedence Computation



Precedence Computation

How to model the precedence-to-the-right rule?



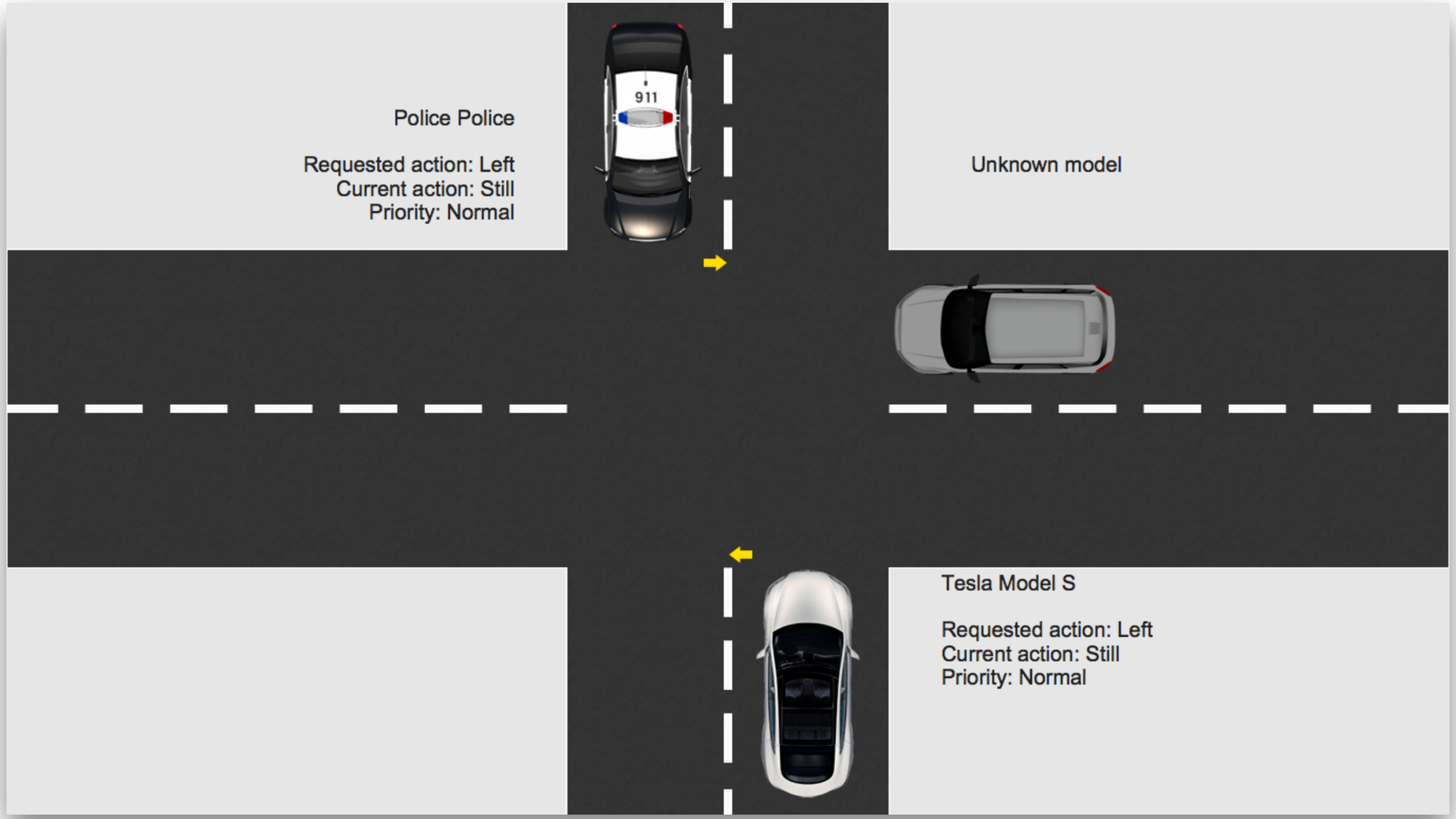
1. Enumerate the lanes counterclockwise, starting from my lane
2. All the cars that satisfy *crossesMyRight* must go before me

$$crossesMyRight(car)$$

$$\iff$$

$$lane(car) \in]1, dest(me)] \wedge dest(car) \geq dest(me)$$

Demonstrator: Monitor



Monitor message format

```
# ARDUINO-MONITOR MESSAGE FORMAT

## COMMON MESSAGE FORMAT

Size: >= 1 Byte
TP...P\n

|   |  FIELD NAME       | DIM |
|---|-------------------|-----|
| T |  MessageType      | 1B  |
| P |  Payload          | any |

 FIELD              VALUE       DESCRIPTION
 MessageType:       I           info message
                    l           sampled data from leftmost receiver (for debug)
                    f           sampled data from front receiver (for debug)
                    r           sampled data from rightmost receiver (for debug)
                    L           frequency spectrum message from leftmost receiver
                    F           frequency spectrum message from front receiver
                    R           frequency spectrum message from rightmost receiver

 Payload:           custom      Data specific to the current message type (see detailed specs below)

This is the format which is used by all the messages.


## INFO-MESSAGE

Size: 26 Bytes
TABCCCCCCCCDDDDDDDDEEEFGH\n

|   |  FIELD NAME       | DIM |
|---|-------------------|-----|
| T |  MessageType      | 1B  |
| A |  RoadID           | 1B  |
| B |  IsEmpty          | 1B  | // Whether the RoadID side of the road is empty
| C |  Manufacturer     | 8B  |
| D |  Model            | 8B  |
| E |  Orientation      | 3B  |
| F |  Priority         | 1B  |
| G |  RequestedAction  | 1B  | // The action the car wants to do
| H |  CurrentAction    | 1B  | // The action the car is doing to cooperate with the network

FIELD               VALUE       DESCRIPTION
MessageType:        I           info message

RoadID:             M           my road
                    L           road to my left
                    A           road ahead
                    R           road to my right

IsEmpty             1           true
                    0           false

Manufacturer:       Vlkswagn
                    Police␣␣
                    Tesla␣␣␣
                    ␣␣␣␣␣␣␣␣    (sent when unknown)

Model:              Beetle␣␣
                    Police␣␣
                    Model S␣
                    ␣␣␣␣␣␣␣␣    (sent when unknown)

Orientation:        [0..360]    degrees counterclockwise

Priority:           0           None
                    N           regular car
                    Y           priority car (ambulance, police)

RequestedAction,
                    0           None
                    L           turn left
                    A           go straight ahead
                    R           turn right

CurrentAction:      0           None
                    S           Stay still
                    L           turn left
                    A           go straight ahead
                    R           turn right
```

Inspector

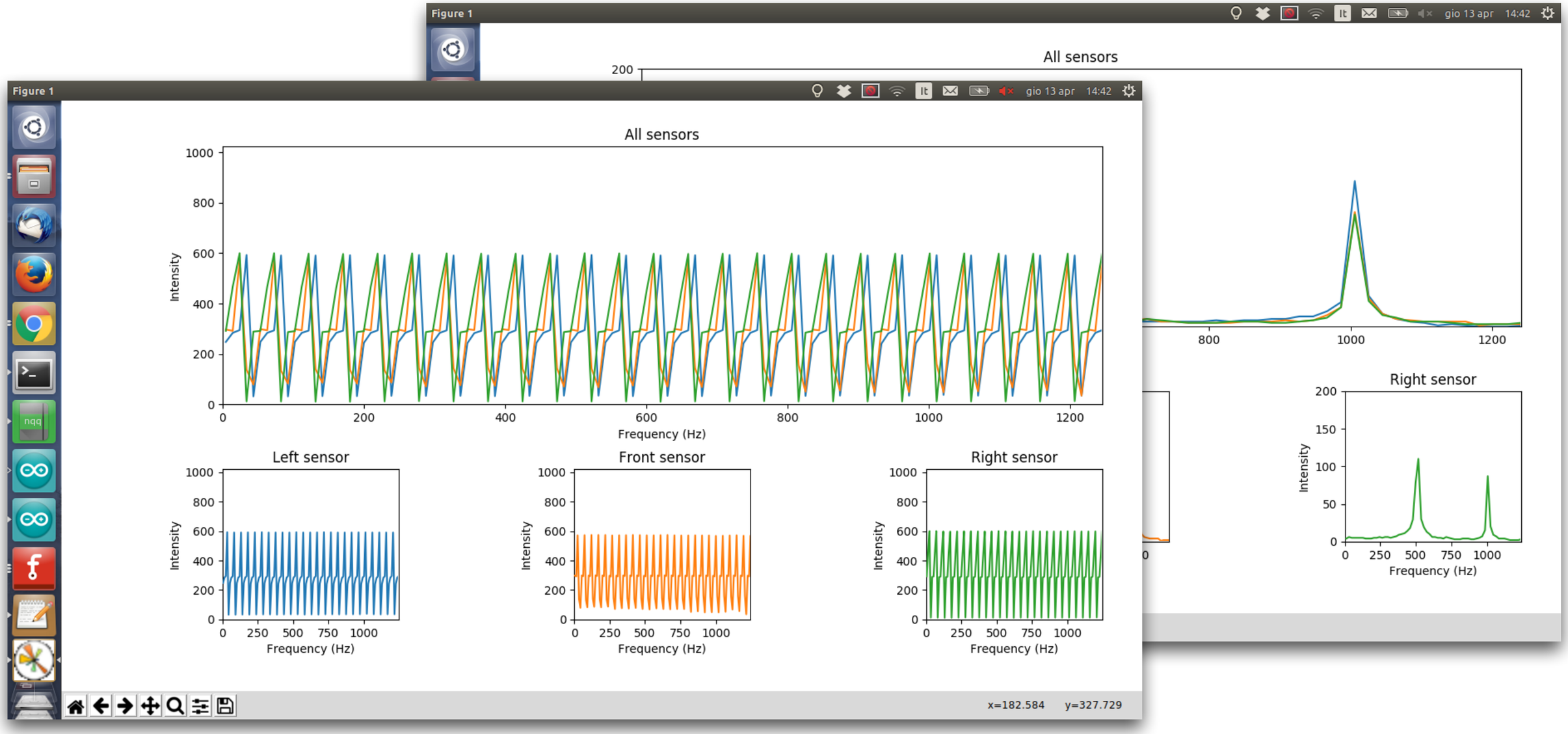


Figure 1
gio 13 apr 14:42
All sensors
Intensity
Frequency (Hz)
Left sensor
Front sensor
Right sensor
x=182.584 y=327.729

Inspector

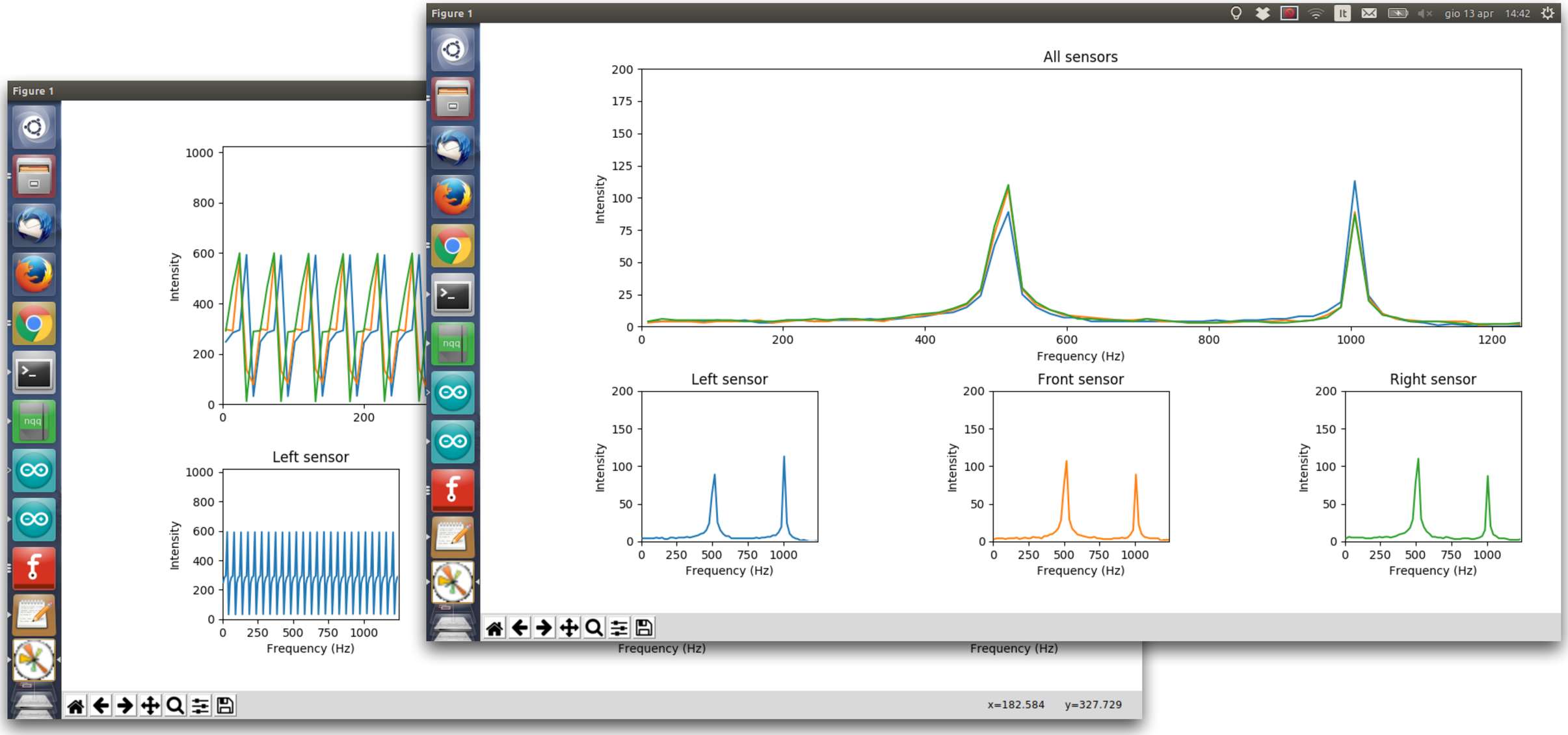


Figure 1
gio 13 apr 14:42
All sensors
Intensity
Frequency (Hz)
Left sensor
Front sensor
Right sensor
Left sensor
Intensity
Frequency (Hz)
x=182.584 y=327.729

Inspector message format

SAMPLED-DATA-MESSAGE

Size: 259-652 Bytes (Assuming FFT_N = 128 = number of samples)

```
TM;A,A,...,A,A\n
   |_________|
        128
```

A,A,...,A,A is the sensor data relative to the IR receiver

	FIELD NAME	DIM	Notes
T	MessageType	1B	one of `l`, `f`, `r`
M	SamplingPeriod	1-10B	(us) the sampling period used in the X-axis of the plot
A	SampleValue	1-4B	values go from 0 to 1023
,	SampleSeparator	1B	separates each pair of samples
;	HeaderSeparator	1B	separates the header from the data

FREQUENCY-SPECTRUM-MESSAGE

Size: 131-396 Bytes (Assuming FFT_N = 128 = 2 * number of bins)

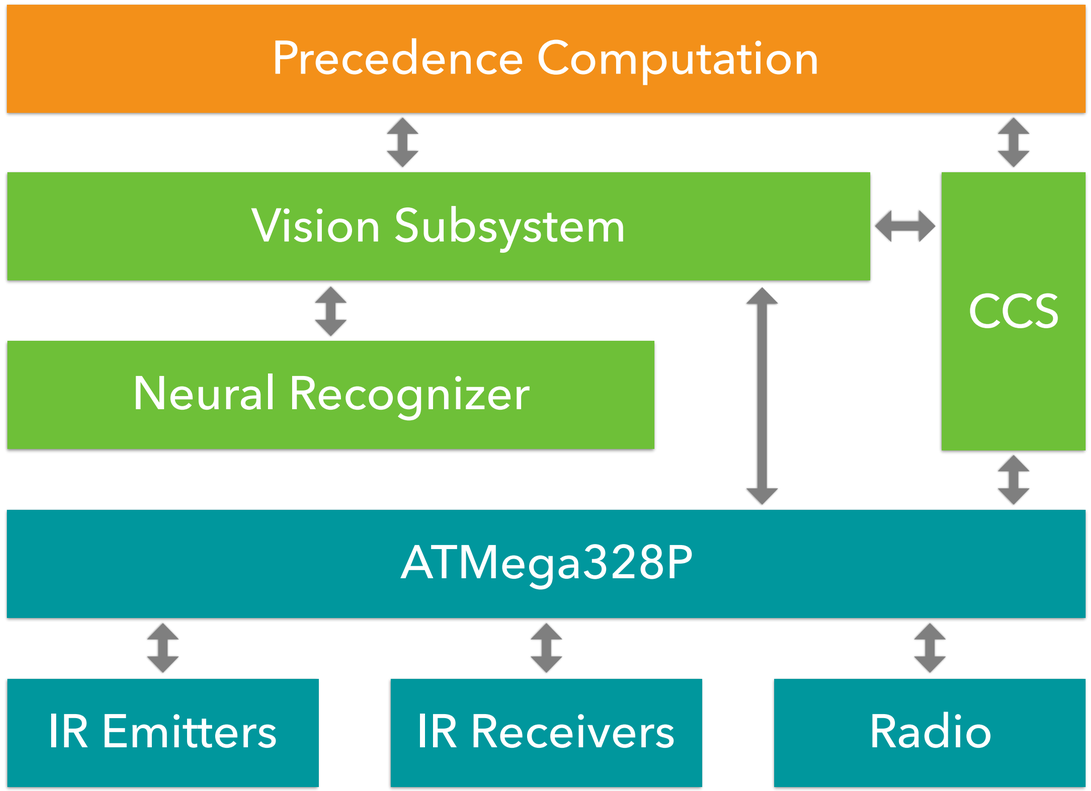
```
TM;A,A,...,A,A\n
   |_________|
        64
```

A,A,...,A,A is the sensor data relative to the IR receiver

	FIELD NAME	DIM	Notes
T	MessageType	1B	one of `L`, `F`, `R`
M	SamplingPeriod	1-10B	(us) the sampling period used in the X-axis of the plot
A	BinFreqIntensity	1-5B	
,	BinsSeparator	1B	separates each pair of bins
;	HeaderSeparator	1B	separates the header from the data

Each bin represents the intensity of a frequency range.
The frequencies go from 0 to samplingFrequency / 2 (where samplingFrequency = 1 / samplingPeriod * 1000000).
Bin i represents the range [i * sampling_frequency / 128, (i+1) * sampling_frequency / 128].

Conclusions



Resource Constraints

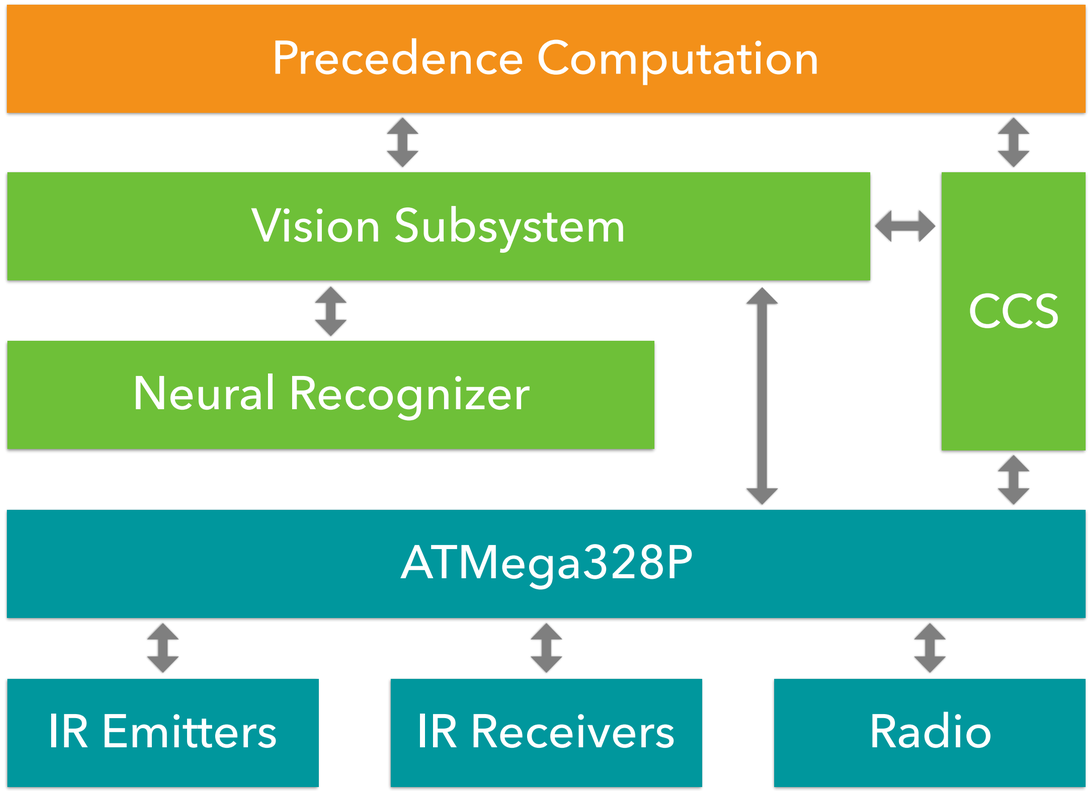
2K of memory and 16MHz

- Emission of IR frequencies
- Sampling with three sensors

} ***real-time***

- FFT computation
- Precedence computation
- Message writing on the Serial Port
- Radio communication with CCS protocol
- Neural Network

Conclusions



Possible Improvements

CCS:

- Listen to the channel before sending messages to reduce collisions.

Vision subsystem (toward moving cars):

- Increase the number of IR leds to detect all the orientations.
- Measure distance between cars

Demo

1. Detection of a single vehicle "jumping" between lanes
2. Basic precedence computation with two vehicles
3. Full precedence computation with three vehicles: