Embedded System Programming Project

Schema

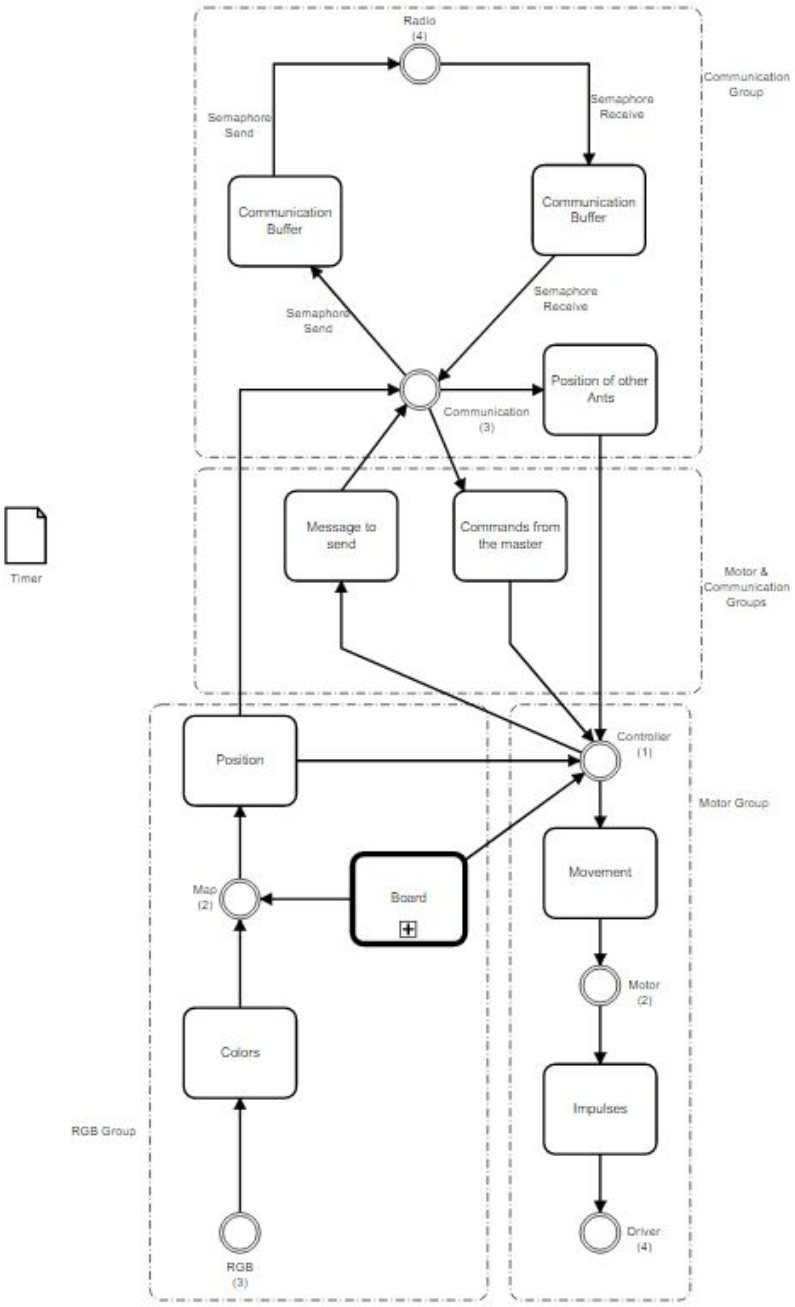

Figure 1: Main Schema

In order to correctly read the diagram, the arrows are placed in a specific way:

- If the arrow **goes into** the communicating element, it means that the task is performing a **write** operation
- If the arrow **goes out** of the communicating element, it means that the attached task is performing a **read** operation

Each circle represents a task. A task is an asynchronous set of instructions that has states, properties and a priority. The priority is represented with a number inside round brackets. The priority is defined as following:

- **4**: Very High
- **3**: High
- **2**: Intermediate
- **1**: Low

Each rectangle represents a communication object. This object can be composed by any type of variables and an indefinite number of them.
Before listing all the communicating objects, we need to introduce some concepts:

- **Circular List**: if the list is full, it will be overwritten starting again from the first element and going forward as usual
- **Hash Map**: list that store items in key/value pairs. The key has to be unique
- **Queue**: list in which the first element to be popped is the last one inserted
- **Semaphore**: technique that allows to control and/or synchronize the access of a shared resource

This is a list containing all the communicating objects and what they are:

- **Colors**: circular list of 10 elements
- **Board**: hash map. Can only be read!
- **Position**: circular list of 20 elements
- **Communication Buffer**: queue of 1 element with 2 semaphores: one for receiving messages from the environment; one for sending messages to the environment
- **Position of other Ants**: vector with 16 elements at max. It is ordered via the ants *id* in ascending order
- **Message to send**: queue of 1 element. If it is empty, then it means that the message is composed by the position of the ant

- **Commands from the Master**: queue of 1 element
- **Movement**: list of 10 elements with a semaphore. It is used in case there are changes to be made to the current path
- **Impulses**: list of 10 elements

Timer is a special communicating object, since it is global, which means that every task can read it. Its value starts with Arduino and updates constantly. Finally, each group has to develop tasks and communicating objects inside the corresponding dashed line. Note that "Message to send" and "Commands from the master" are used by both *Communication* and *Motor* group

RGB

Sensor Information

- **Model**: TCS34725
- **Dynamic Range**: 3.800.000 : 1
- **Sleep State (Consumption)**: 2,5 µA
- **Wait State (Consumption)**: 65 µA
- **Wait State (Time)**: 2,4ms to 7s
- **Communication Protocol: I2C**
  - Data Rate: 400 kbit/s
  - Input Voltage: VDD or 1,8 V Bus
- **Ports**
  - LED: Led Pin
  - INT: Interrupt Port
  - SDA: Serial Data Port (I2C)
  - SCL: Serial Clock Port (I2C)
  - 3V3: 3V input current
  - GND: Ground
  - VIN: Additional power port
- **Photodiode Array**: 3 X 4, composed by red-filtered, green-filtered, blue-filtered, and clear (unfiltered) photodiodes.

Task Actions

- Driver for reading colors

Interaction

Writes

It writes onto the following communicating objects:

- **Colors**: <red,green,blue>, timestamp

Additional Information

It needs to send only the variation of color. This means that, if the color hasn't changed, it doesn't update "Colors"

Map

Sensor Information

/

Task Actions

- Map the colors into a value
- Determine the direction

Interaction

Reads

It reads from the following communicating objects:

- **Colors**: <red,green,blue>, timestamp
- **Board**: <position, <red,green,blue> >

Writes

It writes onto the following communicating objects:

- **Position**: position, angle of rotation, timestamp

Note: the default value for the "angle of rotation" is zero and it indicates the North

Additional Information

Not every position is mapped into the board, but only few points. The task has to define a function that, given a certain point, it can calculate the nearby ones. Moreover, it has to update "Position" only if it has changed since the previous reading

Communication

Sensor Information

/

Task Actions

- Send data
- Receive and decode data
- Token ring management

Interaction

Reads

It reads from the following communicating objects:

- **Position**: position, angle of rotation, timestamp
- **Message to send**: message
- **Communication Buffer**: master command; position of another ant; message from another ant

Note: the default value for the "angle of rotation" is zero and it indicates the North

Writes

It writes onto the following communicating objects:

- **Communication Buffer**: "Message to send" == null ? "id_ant, position, angle of rotation, timestamp" : "message"
- **Position of other Ants**: id_ant, position, angle of rotation, timestamp
- **Commands from the Master**: master command

Additional Information

The task has to manage the token ring configuration: it must know when it is his turn to communicate with the others.
Decoding the incoming messages should be a fast action, since it is necessary to know the location of other ants as quickly as possible.
When sending the position of the ant, the task has to take the last position, which is in the head of "Position"

Radio

Sensor Information

- **Model**: HC-12
- **Frequency radio signal**: 433.4 MHz to 473.0 MHz
- **Total Channels**: 100
- **Stepping between channel**: 400 Khz
- **Transmitting Power**: from -1dBm (0.79mW) to 20dBm (100mW)
- **Receiving Sensitivity**: -117dBm (0.019pW) to -100dBm (10pW)
- **Table of Recommended Values**

Receiver Sensitivity	**Over-the-Air Baud Rate**	**Serial Port Baud Rate**
-117 dBm	5000 bps	1200 bps
-117 dBm	5000 bps	2400 bps
-112 dBm	15000 bps	4800 bps
-112 dBm	15000 bps	9600 bps
-107 dBm	58000 bps	19200 bps
-107 dBm	58000 bps	38400 bps
-100 dBm	236000 bps	57600 bps
-100 dBm	236000 bps	115200 bps

- **Voltage**: 3,2V to 5,5V
- **Distance**: 1Km on the open air
- **Ports**
  - VCC: Input current
  - GND: Ground
  - RXD: Input, weak pull-up
  - TXD: Output
  - SET: Input, internal 10k pull-up resistance. For control pin
  - ANT: Antenna

Task Actions

- Driver for sending data
- Driver for receiving data

Interaction

Reads

It reads from the following communicating objects:

- **Communication Buffer**: message

Writes

It writes onto the following communicating objects:

- **Communication Buffer**: message

Additional Information

Every ant must specify the *STX* (Start of Text) and *ETX* (End of Text) when sending and receiving a message
After performing an operation, the next task must be "Communication"

Controller

Sensor Information

/

Task Actions

- Move the ant based on multiple factors
- Execute commands received from the master

Interaction

Reads

It reads from the following communicating objects:

- **Position**: position, angle of rotation, timestamp
- **Position of other ants**: id_ant, position, angle of rotation, timestamp
- **Commands from the Master**: master command
- **Board**: <position,<red,green,blue»

Writes

It writes onto the following communicating objects:

- **Movement**: angle of rotation, speed

Note: the default value for the "angle of rotation" is zero and it indicates the North

Additional Information

This is the slowest task, since it has to do a lot of calculations.
All the algorithms involved must be as efficient as possible, without compromising too much the precision

Motor

Sensor Information

/

Task Actions

- Map the information of the movement into impulses

Interaction

Reads

It reads from the following communicating objects:

- **Movement**: angle of rotation, speed

Writes

It writes onto the following communicating objects:

- **Impulses**: impulse

Note: the default value for the ”angle of rotation” is zero and it indicates the North

Driver

Sensor Information

Motor

- **Model**: sy35st26-0284a
- **Working Temperature**: -20 ~ +50
- **Number of Phases**: 2
- **Max Radial Force**: 28 N
- **Max Axial Force**: 10 N
- **Steps per Revolution**: 200
- **Step Angle**: 1,8°
- **Torque**: 650 g-cm / 6,3743225 N-cm
- **Voltage**: 7,28 V
- **Current**: 0,28 A
- **Output Shaft (type)**: D, with a section flattened by 0,5 mm
- **Output Shaft (length)**: 21 mm
- **Output Shaft (diameter)**: 5 mm

Driver

- **Model**: Adafruit Motor/Stepper/Servo Shield for Arduino v2 Kit - v2.3
- **Driver**: TB6612FNG MOSFET
- **Compatibility**: 4 DC Motors or 2 Step Motors
- **Protocol**: I2C, 7-bit addresses between 0x60-0x80, selectable with jumpers
- **Stack Capability**: 5 address-select pins, up to 32 stackable shields. Up to 128 DC Motors or 64 Step Motors
- **Motor Voltage**: 4,5V DC to 13,5V DC
- **Channel Current**: 1,2 A. It can use up to 3A for 20ms

Shaft

- **Model**: Pololu Universal Aluminium Mounting Hub for 5mm Shaft, M3 Holes
- **Material**: Aluminium
- **Shaft Diameter**: 5 mm
- **Mounting Hole Size**: M3

Wheels

- Model: Pololu Wheel for FEETECH FS90R Micro Servo
- Shaft Diameter: 5 mm
- Dimension: 60 mm

Task Actions

- Move the ant by rotating the wheels

Interaction

Reads

It reads from the following communicating objects:

- **Impulses**: impulse