LABORATORIO DI INGEGNERIA DEI SISTEMI SOFTWARE

Introduction

Our motto:
there is no code without a project, no project without problem analysis and no problem without requirements.

Requirements

Requirements are defined in RadarSystem.html

Requirement analysis

MAIN GOALS:

1. clarify the **meaning** of the *names* and of the *verbs* included in the requirement text given by the customer
2. (informally) define a first set of **functional TestPlans**

DOMAIN GLOSSARY:

1. **radarGui**: software component that show distance from Sonar.
2. **Sonar**: hardware component provided by client , detect distance from Sonar , name of the model is ``HC-SR04`` , the distance is read by software entity.
3. **RadarDisplay**: GUI for the radarGui , it is commanded by sending coordinates.
4. **ledAlarm**: hardware\software component that is on if distance detect by Sonar
5. **Led**: hardware component provided by client . The component has 2 state on/off ,it can be commanded by gpio port.

6. **DLIMIT**: threshold value for turn on\off led.

Ambiguity of the requirements: the requirement that the led turns on when the distance is less than DLIMIT implies that if the distance subsequently increases beyond DLIMIT the led stays on. The requirement is interpreted as: the led turns on when the distance is less than DLIMIT and turns off when the distance is greater than DLIMIT.

TEST PLAN:

1. **Test 1**:start with led off , move an object close to sonar ,when object is less than DLIMIT and led is on test is ok

Problem analysis

MAIN GOALS:

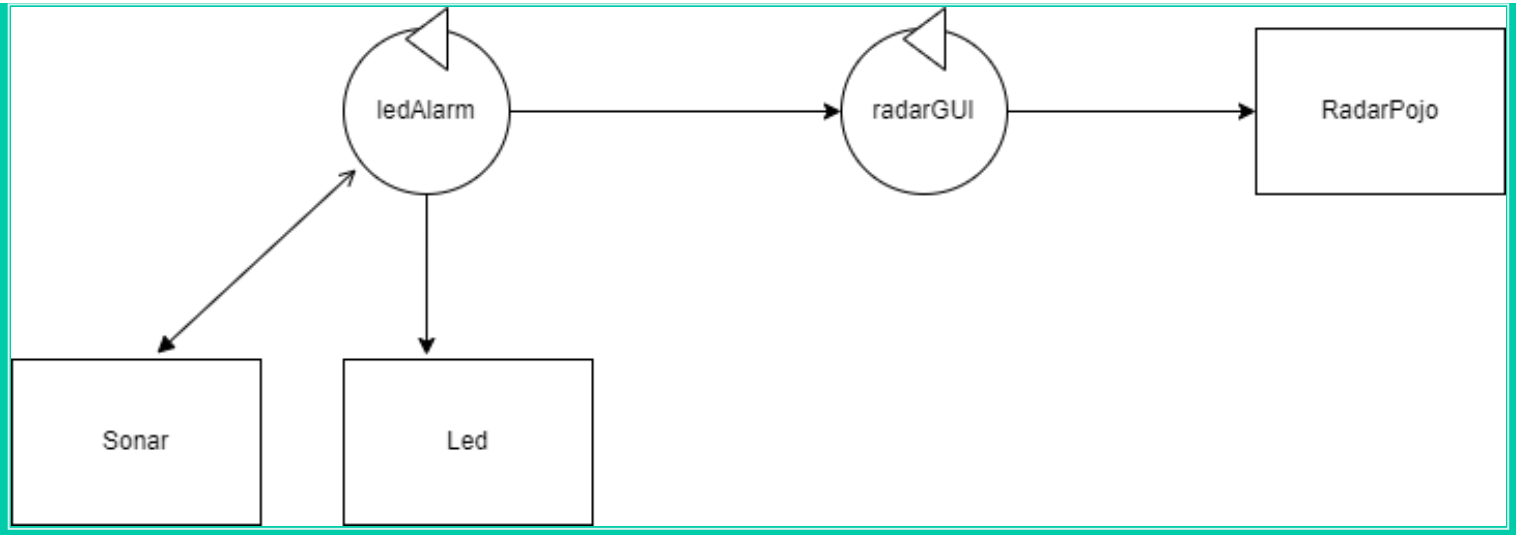
1. identify the main problems involved by the requirements and the most appropriate (software) technologies to adopt
2. evaluate the **abstraction gap** and give a **quantitative measure of the effort/resources** necessary to build the system
3. define (a **model** of) the **logical architecture** of the system
4. refine the set of **functional TestPlan**
5. (with reference to **SCRUM**) define a (first) **product backlog** and a possibile set/sequence of **SPRINT**

PROBLEM ANALYSIS:

The following table shows the requirements and broken down into architectural dimensions. Dist is the value read by the radar which is transmitted to the rest of the system.

Requirements	Component	Structure	Comunication	Behaviour
RadarGui	RadarPojo	PC	Input Dist	Show Dist
LedAlarm	Sonar HC-SR04	Rasberry	Output Dist	Read Dist
LedAlarm	Led	Rasberry	Input Dist	f(Dist) => on\off

This makes it clear that there are two logical entities ledAlarm and radarGUI that communicate from ledAlarm to radarGUI.



Logical Architecture

As far as radarGUI is concerned, it is satisfied by an entity that contains radarPojo which receives a message containing the value of Dist is called Radar.
The time between two updates of the Dist value on the radar is TRADAR.
For the ledAlarm side, since for now the sonar and the led can be on the same node, the value of f (Dist) must not leave the node.
To satisfy a property of independence between the radar and the sonar \ led pair it is necessary to add a communication entity with the sonar \ led pair which is called Sonar.

At this point the abstraction gap is clear: how to read the sonar distance? , how to transmit the distance value? , how to control the LEDs?
Technological problems follow: how to make different technologies interact? , is it possible to reduce the number of technologies used in order to reduce the number of release entities? is it possible to reduce the number of messages? what's the trade-off with code reusability?
From the Sonar entity to the Radar there are no significant abstraction gaps to fill.

To solve the sonar reading requirement a Sonar.c library has been provided which writes to standard output, this library does not meet the requirement for communicating with the rest of the system.
The problem of reading and communicating implies that a technological solution that satisfies both requirements is preferable.
The designer prefers the use of python with its gpio and mqtt libraries to this library, allowing to satisfy the requirement of

reading and communicating with the rest of the system.
The entity that reads and communicates is called SonarRobot.
The time between two sonar readings is called TSONAR.
SonarRobot publishes on mqtt server (P.Dist) and Sonar subscribes (S.Dist).

A library is provided to control the LEDs which it calls GPIO shell library commands.
The problem of communication and reduction of the number of technologies used implies that also in this case the designer uses python code to be integrated in SonarRobot to control the LEDs and minimize the release files.
At this point the problem arises of choosing between an architecture that favors reusability, more expensive in terms of released entities, or a more monolithic but less expensive approach in terms of released entities.
The designer chooses not to maximize the reusability (NMR) of the code to reduce the released components and the number of messages. The other choice is to have a Led entity that commands via mqtt a python entity (LedRobot) that commands the leds directly with python code or calls the scripts provided , or in alternative led use mqtt to comunicate to PC and command led in one of the previous ways.
The choice, maximum reusability (MR), would imply that LedRobot publishes (P.Led) on the mqtt server and Led subscribes (S.Led).
SonarRobot also manages access to the file system to obtain the DLIMIT value used as a threshold to control the on / off values of the LED.
It should be noted that the choice made is completely consistent with the requirements as neither reusability nor the communication of information regarding the management of the LEDs outside the entities defined in the ledAlarm requirement is required.
The advantage of the choice made is also to have fewer messages exchanged in the system as opposed to the choice of greater reusability.

In order to decouple Sonar and Radar, a Controller entity is introduced which requests the Dist value from Sonar and sends the Dist value to Radar.
Since it is preferable to have TSONAR other than TRADAR, as they are different components whose optimal values are to be calibrated during the development phase, the Controller entity must manage the TRADAR update, so it will contain a timer.
Controller asks Sonar for the Dist value which responds with the requested value with TRADAR period, and consequently sends it to Radar.

FRADAR = 1 / TRADAR
FSONAR = 1 / TSONAR

alpha = messages sent to broker / sonar readings

Defined NLED the number of times the LED is commanded, the choice made indicates that the number of messages exchanged is: 3 * FRADAR + S.Dist + alpha * 2 * FSONAR, on the contrary the choice with the Led entity is the number of messages traded is: 3 * FRADAR + S.Dist + alpha * 2 * FSONAR + S.Led + 3 * NLED.

Architecture	Number of message	Number of deployed entity
No Max Reusability	3*FRADAR + S.Dist + alpha*2*FSONAR	SonarRobot.py + CtxPc
Max Reusability	3*FRADAR + S.Dist + alpha*2*FSONAR + S.Led + 3*NLED	2*sh + Sonar.c + Led + Sonar + CtxPc*

CtxPc contains the entities to be released on the pc node, with CtxPc * we mean that the MR choice implies a context of minor elements.

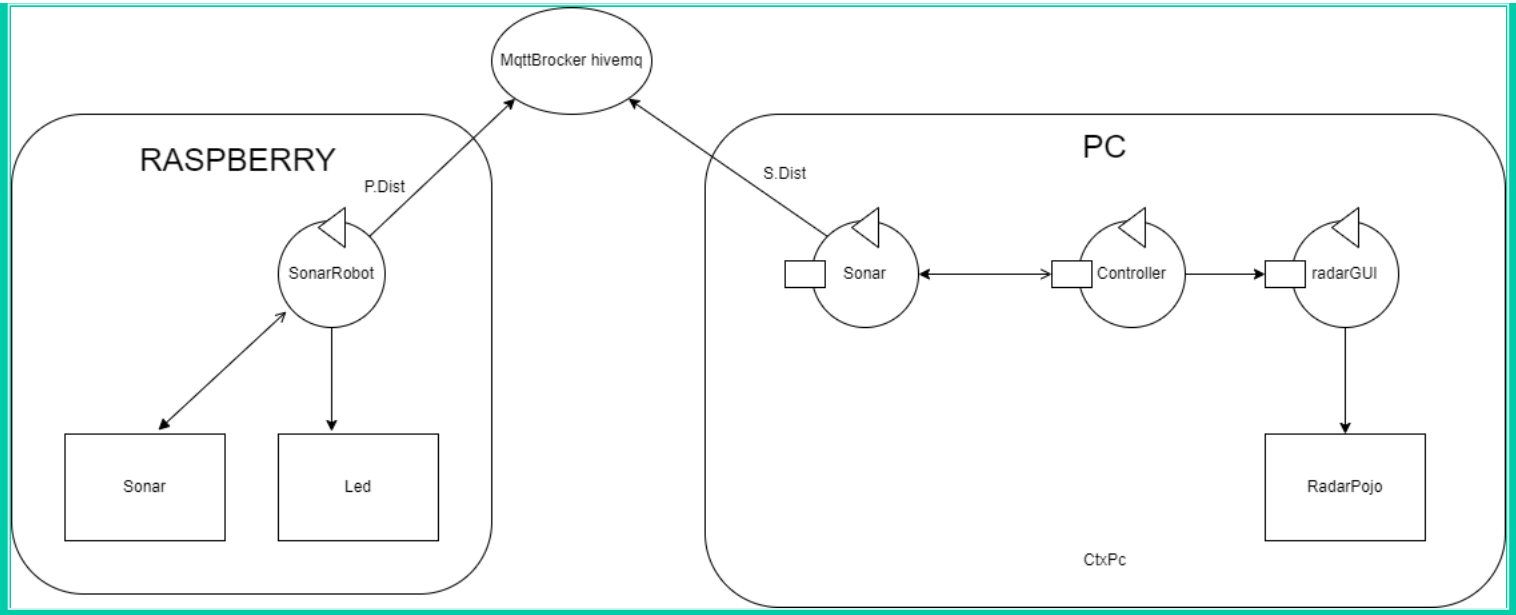
WARNING: expressions like *'we have chosen to ...'*, *'I decided ...'*, etc. are **forbidden** here.
Rather, this section should include sentences like ***'this (aspect of the) problem implies that ...'*** or ***'the usage of this (legacy) component requires that ...'***, etc.

Project

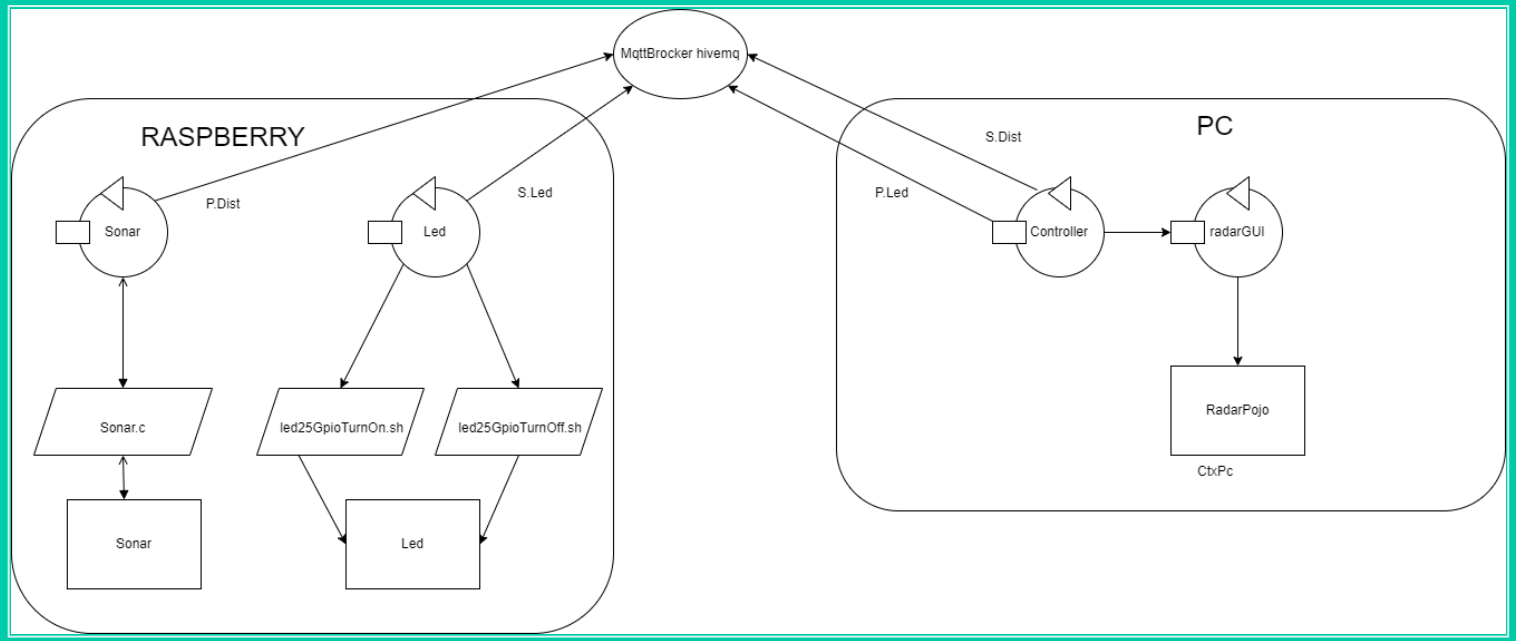
MAIN GOALS:

1. by starting from the **logical architecture** of the system, define the **concrete/k> architecture of the system and the behavior of each component .**

PROJECT:



Project Architecture



Alternative Project Architecture

By studentName email: ugo.marchesini@studio.unibo.it