# 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

**es1**: Turn on/off a *led* using a button (ON/OFF). The system should be a distributed system, in which the Button and the Led run on a *different computational supports*, e.g. a Conventional PC, a RaspberryPi or Arduino.

## Requirement analysis

| Term | Meaning |
|---|---|
| Turn | Logical action that imply transition from one state to another in a binary state space |
| Led | Output, programmable as a memory mapped device |
| Different Computational Supports | Two, separate system. One contains the led, the another contains the button |

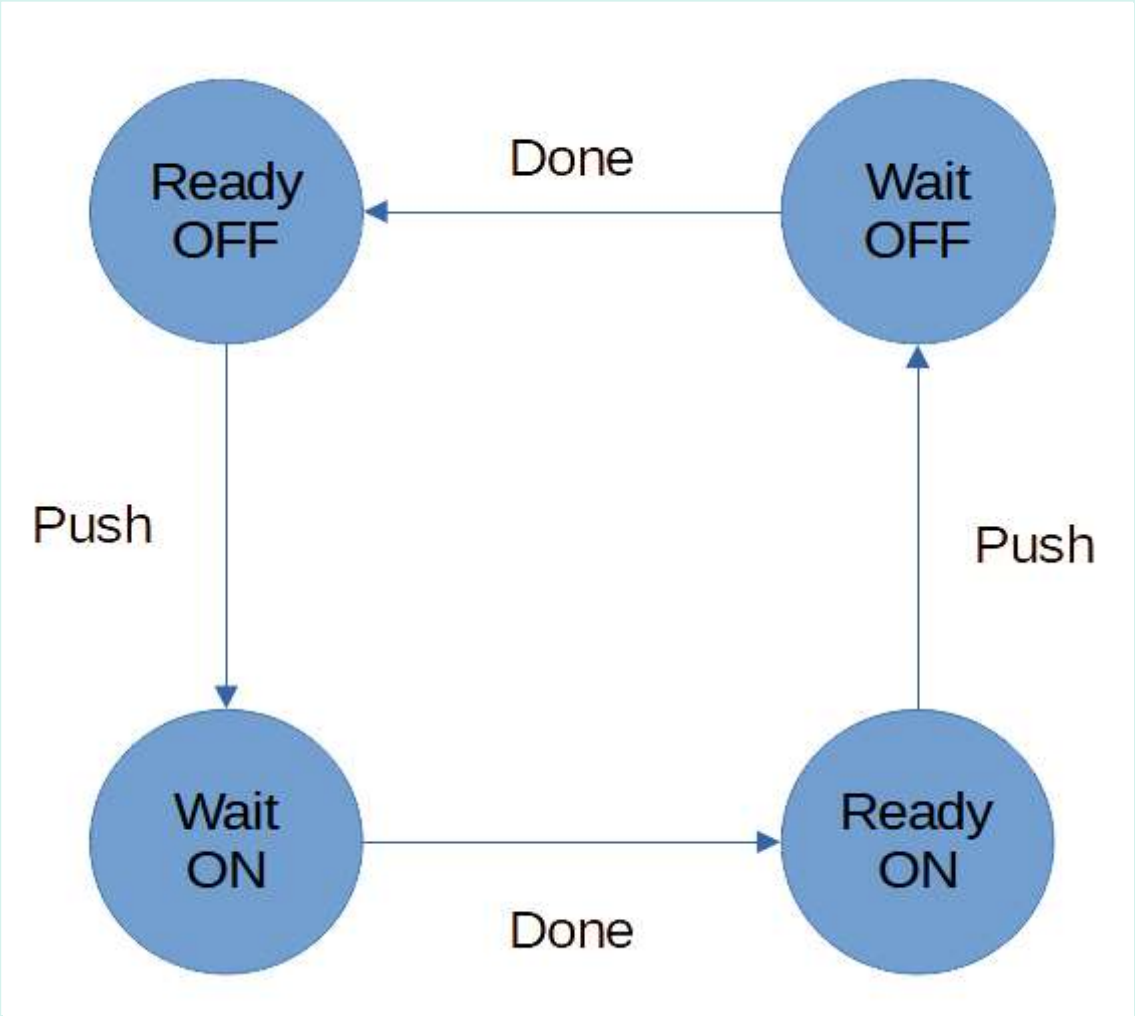
### User Story

An user push the button while the led is off. The led turn on. The user press the button again and the led turn off.

## Problem analysis

We assume that there is only one client and one server (per description of the problem). There are two actors: the local actor (client) and the remote actor (server).

We identify the states of the client actor and its transitions on appropriate messages. There are 4 states.




The server actor is specular, but receives only from the net. The protocol is composed of 3 messages:

1. TOGGLE
2. ON_DONE
3. OFF_DONE

In waiting states the system refuse to accept user input (and it provides this information to the user). To ease the implementation we will use QActor, and so we required it as a dependecy.

The logical architecture of the system is so composed:

1. An user interface with one button
2. The command "pushed button" is sent to the local actor
3. The local actor decide if it should ignore the input or comply
4. If the actor complies it sends the TOGGLE message to the remote actor
5. The local actor go to a state that is insensitive to user input and sensitive to the expected response ON_DONE/OFF_DONE
6. The remote actor complies and report ON_DONE/OFF_DONE
7. When the response come the actor transition back in a state of readyness for user's command

# Test plans

To verify the correct behavior of the implementation, the actor's state should be observable. The test plan is described below:

- The led is put off
- Send a message to the actor faking user interaction
- Listen to incoming messages from the network
- If an ON_DONE message is received the test is successful

---

By Alessio Ferri email: alessio.ferri@studio.unibo.it