Communication Protocols

Embedded systems 2

Authors: Veselin Atanasov, Toon Spitshuis

Teachers: Hans van Heumen

Date: 18-03-2024

Abstract

The goal of this assignment (Heumen, 2023) was to create a traffic light system, that makes use of a robust and well-designed communication protocol. The assignment also functioned as an opportunity to learn best practices regarding communication protocol designs.

Contents

Introduction

The task in this assignment (Heumen, 2023) was to create a flexible robust traffic light system for a road construction company. The system should consist of a distributed embedded system that makes use of the Arduino microcontroller unit. The main focus was on a robust communication protocol for the traffic light system.

Procedure

The first step of producing a satisfactory product is the design. In order to create an effective solution, a clear problem statement is required. In this project the task is to create a traffic light system. The first step is to identify all the possible problems when having a traffic light system consisting of two microcontrollers communicating with eachother. After defining the problems, possible solutions were discussed and agreed upon. The problems are that the communication might fail or that the message is corrupted another problem is possible hardware issues. Most of the issues will be fixed by designing and implementing an effective communication protocol to handle the different cases with the help of an error state. The designing for this project included creating a protocol design, a state diagram(happy and unhappy flow) and a hardware diagram. The design of the hardware includes choosing the right hardware and determining the safety of each component in the circuit.

I_F = 30 mA

U_F at I_F = 2,5 V

$U_R = U - U_F = 5 - 2,5 = 2,5$ V

$I = I_F$

$R = U_R / I = 2,5 / 0,03 = 83,33$

$P_{Tot} = U * I = 5 * 0,03 = 0,15$ W

To have a proper protocol design, the use case should be clear and commands should be defined. More detailed overview of the process of designing can be seen in the design paragraph in the report.

The second step is to implement the design into an actual product. After the hardware diagram is ready and calculations conclude the safety of the circuit the components are placed on a breadboard accordingly. After that the programming of the Arduino can start. For this project the two microcontrollers are connected and communicate using wired serial communication. The first thing to program is the happy flow. The happy flow is a case in which everything goes smoothly and according to plan. In the case of a traffic light the flow of the states is red-green-yellow-red. In this case there also needs to be a period which both lights are red. To achieve this the following order of the lights is used master:red-red-red-green-yellow-red; slave-green-yellow-red-red-red-red. The master changes state based on a timer and sends a command to the slave so the slave also changes its state accordingly. The programming of the happy flow went mostly smoothly the only challenge was to create a state in which the red light on the master is o but the master keeps sending commands. This issue was resolved by using a wait state which has 3 substates: sendGreen, sendYellow, sendRed. Those substates use the same timers as the red, yellow and green states and when the timer elapses the state changes in all of

those substates the light is red. The program for the slave Arduino is pretty simple. It takes the command it receives and executes it. Once the basic communication was covered the programming went relatively easy. The unhappy flow requires the addition of a new state called error. In this state the yellow lights is blinking. If one Arduino is in error state so should the other. The program goes into this state when communication fails or a button on the master Arduino is pressed. To ensure that the communication is working the slave sends ACK message if the command is received and recognized and NACK if it is received but not recognized. If the master Arduino stops receiving ACK or NACK then it goes into error state because the communication has failed. For the slave configuration to go into error state it either has to receive a command from the master or stop receiving messages which means that the communication has failed. Some of the commands from the original protocol design are not used as they are not serving any additional functionality. The commands REQ and CUR are not needed as the slave is only responsible for executing the commands it receives and sending a confirmation. The master needs to know only if the command is received not the state of the slave as the master already dictates the state of the slave.

Design

The traffic light system as can be seen in figure 1 consists of two sub systems, traffic light 1 and 2. One of these subsystems acts as the master and the other as the slave. The master is active and sends commands to the slave and the slave is reactive and sends responds to the master.

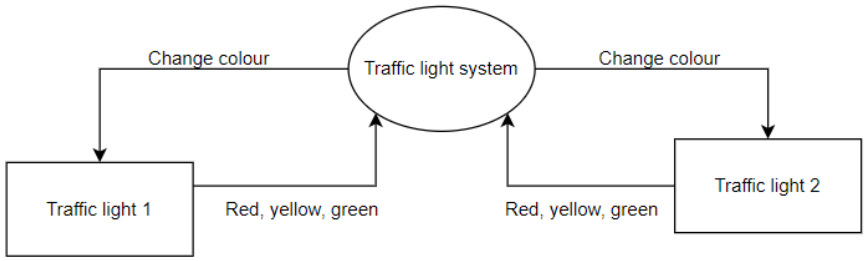

Figure 1

The communication protocol is a crucial part of the project. It gives the tools to handle any situation properly. The communication protocol consists of commands depending on the current situation and the functionality that is required. See the table for commands and further explanations.

Command	**Explanation**	**message**
Acknowledge	Send when msg is received properly and is understood	ACK
Not acknowledge	Send when msg is received in-properly or is not understood	NACK

Request state	Asks for slave's current state	REQ
Current state	Gives current state	CUR#GREEN, CUR#RED, CUR#YELLOW
Force state	Forces slave to state	FORCE#GREEN, FORCE#RED, FORCE#YELLOW
Error	Send to other system when system goes into error mode When received system also goes into error mode	ERROR

To see an example of the communication between the microcontrollers see the sequence diagrams in figure 2 and 3.

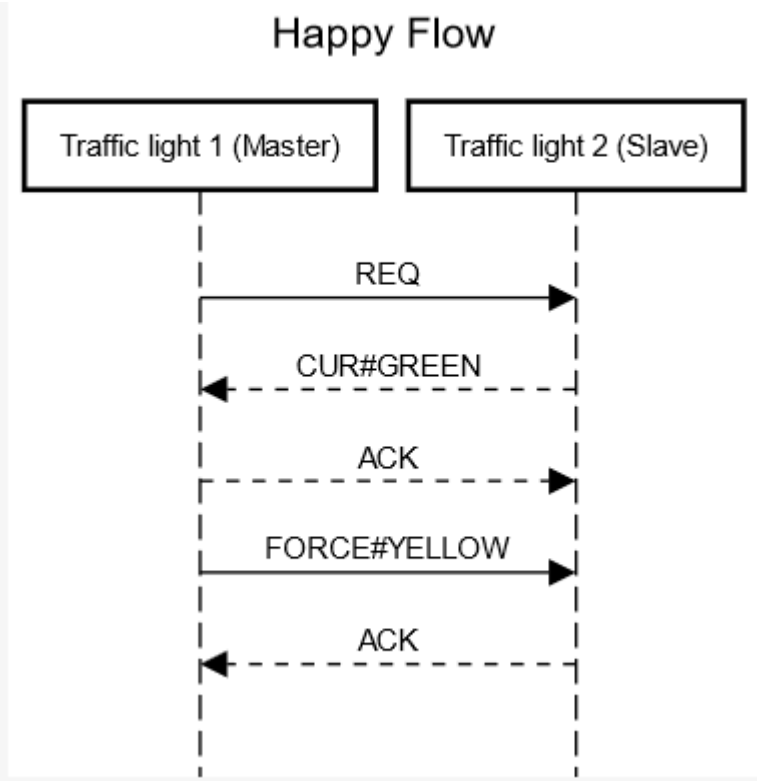

Figure 2

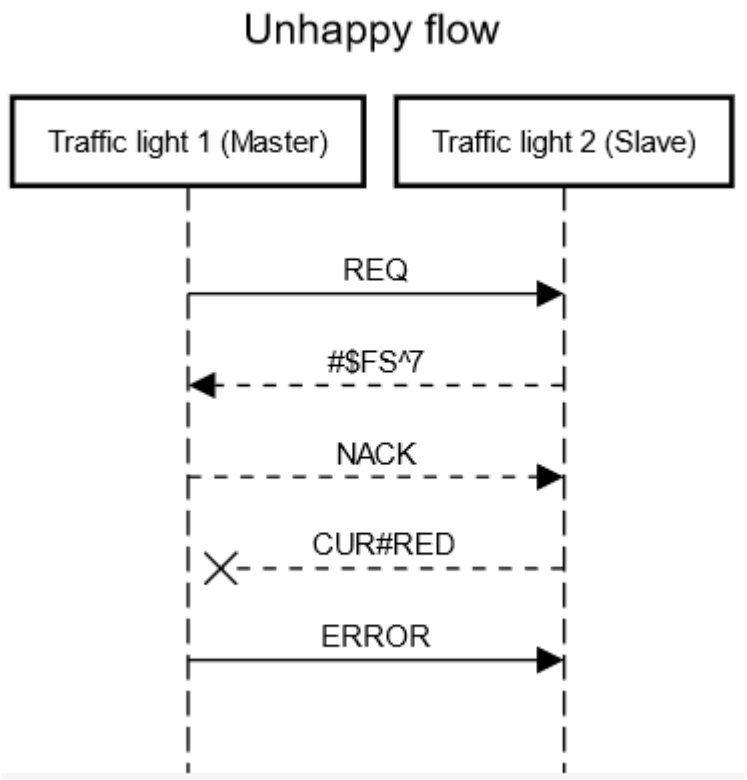

Figure 3

The hardware diagram provides a view of the different components. In this case the leds an the resistors that protect them can be seen. The communication is also showcased. The RX of the master is connected to the TX of the slave and the TX of the master is connected to the RX of the slave. This can be seen in figure 4.

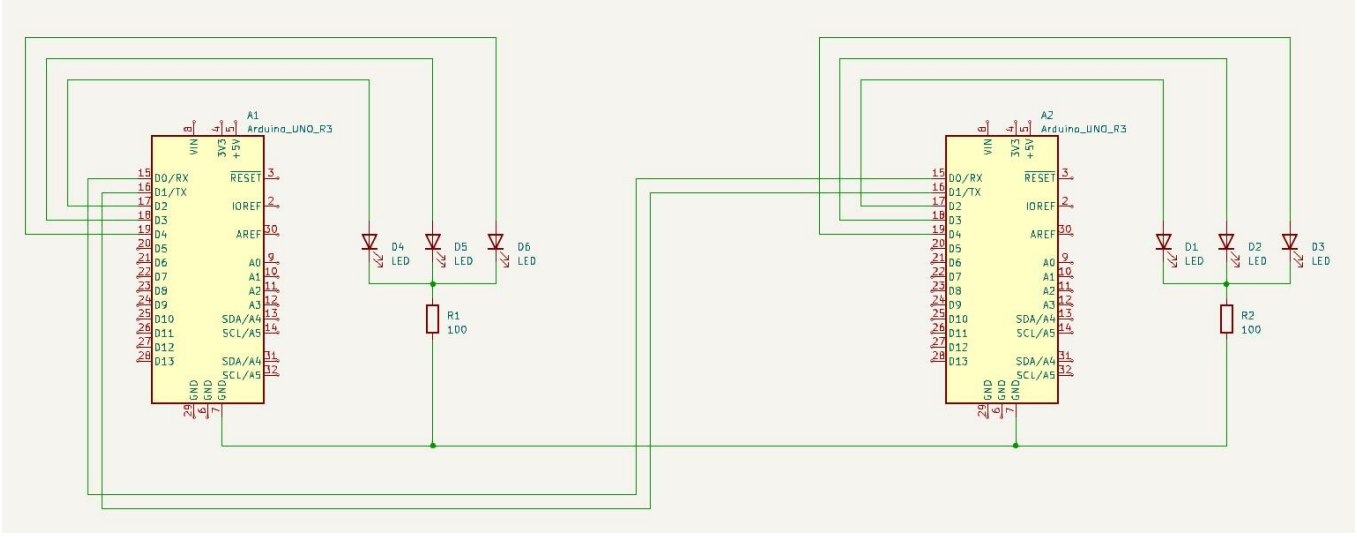

Figure 4

The state diagram is an important part of the design. The state diagram shows the different states and the transition between them(condition to change states and action to perform when state is changed). In this diagram the master changes state based on a timer. When this timer expires the state changes and new messages are being send and the timer is reset. See figure 5 for the happy flow.

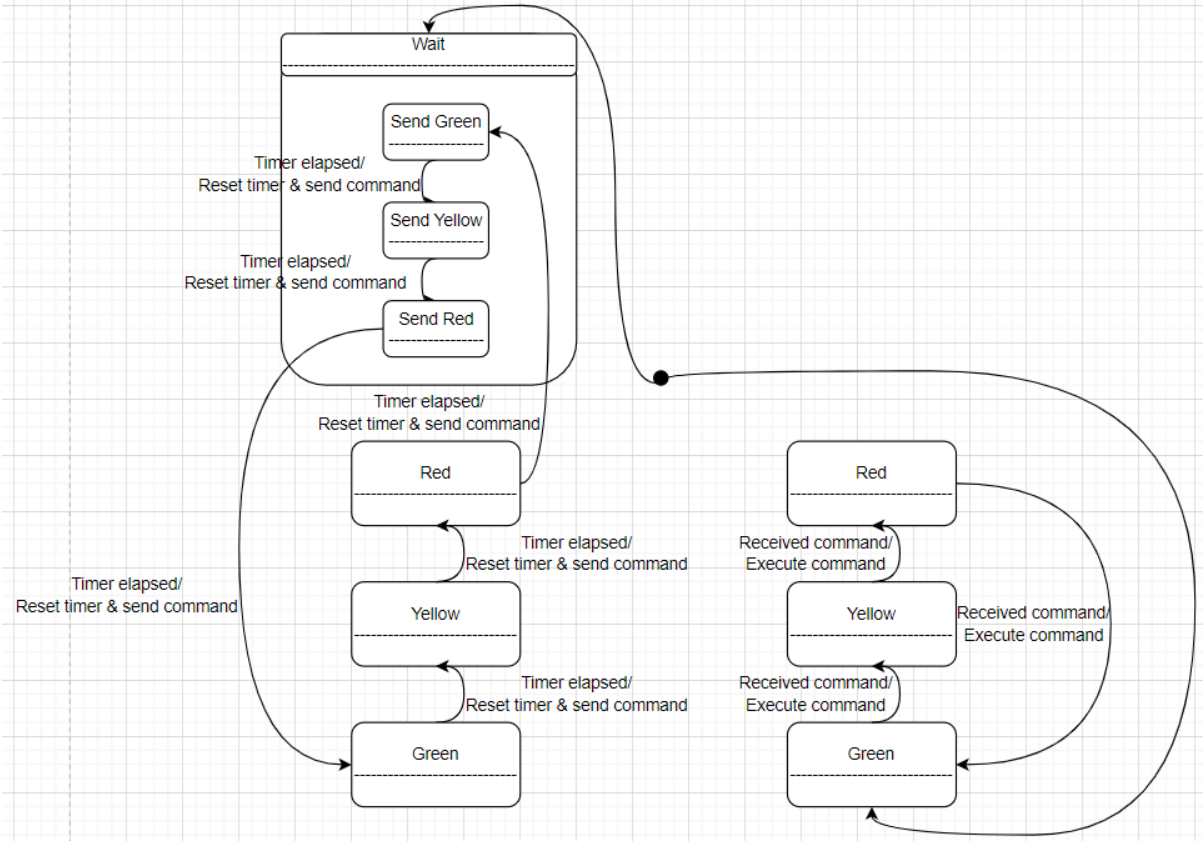

Figure 5

The unhappy flow has an extra state called error. It is used to handle different scenarios in which things do not go as planned. The program can go to the error state from each state. There are two conditions to go to this state. Either the communciation has failed or a physicaly button on the master arduino configuration is pressed. We can go to the initail state from the error state if the button is pressed or communication is restored. The unhappy flow of the state digram is shown in figure 6.

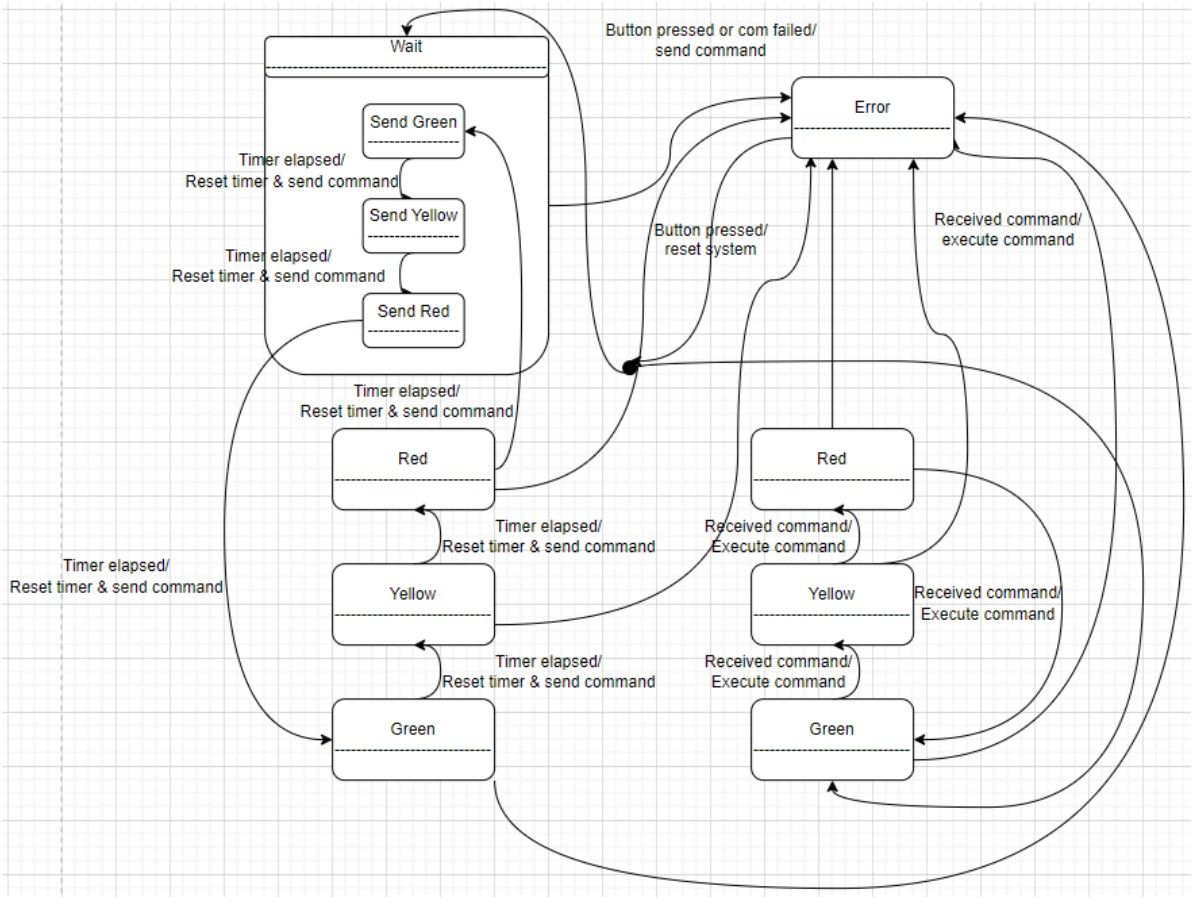

Figure 6

Conclusion

This document showcases the process of designing a traffic light system consisting of two Arduino boards. This document also features all the designs needed to achieve the wanted results and the whole procedure of designing and implementing. The design of a project is one of the first and most important steps of creating a product that meets expectations and fulfills requirements. Having a good protocol design and different supporting forms of visualizing overall structure and concepts in the form of sketches or diagrams is of great help during the implementation process.