**Assignment 2**

Embedded Systems 2

Practicum Assignment 2

February 8, 2023.

**Group 2**

Students: Johnson Domacasse

Tom Koppelmans

# Abstract

This technical report is about applied research of how to create reliable and robust communication protocols via the serial port (UART). A simple sender and receiver of a message are designed such that they are essential for creating a reliable and robust communication protocols. The designs are based on state diagrams. These designs are then tested using a traffic light system to confirm these results. Overall, the results of this research is useful for designing reliable and robust communication protocols over serial connections using a microcontroller.

Contents

[1. Introduction 3](#_Toc131262238)

[2. Procedure 3](#_Toc131262239)

[3. Designs 3](#_Toc131262240)

[4. Conclusions 3](#_Toc131262241)

[5. References 3](#_Toc131262242)

# 1. Introduction

Building and using communication protocols via the Serial object on the Arduino is no complex task. For one you would need a *Master* that sends all the messages. Then you would need a *Slave* on the other side to receive and process all of these messages. Take for example that the Master sends a message that reads: “hello there” to the Slave. The Slave will read this and there could be a command within the Slave programming that states that, if the slave reads “hello there”, It would turn on an LED. All of this happens through the serial object and Arduino. This matter of communication does indeed work, however it is neither reliable, nor robust. Think for example if a wire has been disconnected. A reliable system should be able to handle these situations in the form of failsafe’s.

**We can then ask ourselves the question: “How can we make such a reliable system”?**

In order to this do we need to take into account that every situation or occurrence needs a system state in order to keep working properly. The *happy* state is the state that run perfectly without any inconveniences. However, the *unhappy* state is the state that would deal with these inconveniences. Take for example if a wire has been damaged. A message is corrupted. A message hasn’t been received. These inconveniences can be vast depending on what type of system you are working on.

In this assignment, We will be making use of state-diagrams. This way we are able to discover and handle our entire system. This seems like the ideal solution, since we are working by sending and receiving messages. We make use of coding skills that will not delay the entire system as multiple states are being executed simultaneously.

We implement this by making use of an two Arduino uno boards, two LED traffic light modules and lastly, a few jumper cables.

Initially, Some resistors were also used but within the document we will explain why we don’t use them in the final system.

# 2. Procedure

To start off, we need established what we were trying to accomplish a good design of the already existing traffic lights we see around. We started off by making a simple design of how we thought the system could look like in the happy state. See figure 1. As you can witness, we had a simple system that required two Arduinos. Each of them would turn LEDs both off and on through digital connections. The only difference is that one Arduino, Traffic lights master, would send out commands to the other Arduino, Traffic light slave. The slave Arduino then either sends back acknowledged (ACK) or not acknowledged (NACK).

With this design we got to work on our state diagram designs. See section 3 *Designs*.

## 2.1 Design approach

We made use of a few states and some functions to go within those states in order to realize our desired state diagrams. See table 1 & 2 below. All of these states cover both the unhappy and the happy flow of our system. Each state has their own reasoning for being there.

(**MAKE TABLE 1 AND ADD REASONING BEHIND EACH STATE IN DESIGN APPROACH.)**

**(MAKE TABLE 2 AND ADD FUNCTION OF EVERY FUNCTION IN DESIGN APPROACH.)**

## 2.2 Design implementation

(**TALK ABOUT HOW WE IMPLEMENTED ALL OF OUR STATES WITHIN THE CODE.)**

# 3. Designs

(**FIGURE 1 THAT SHOWS OUR FIRST DESIGN)**

**(FIGURE 2 THAT SHOWS OUR FINAL DESIGN)**

# 4. Conclusions

(**DRAW CONCLUSION FROM THE IMPLEMENTATION)**

**(MENTION ANOMALY (NO NEED FOR RESISTOR))**

# 5. References

(**FHICT LINKS)**

**(TRAFFIC LIGHT MODULE LINK)**

**(LINK FOR RX AND TX COMMUNICATION)**

**(LINK FOR THE ADOC DESIGN)**

**(REFERENCE DR. HILDERINKK FOR INSPIRITATION)**