EITF45 - Computer Communication
Lab 1 - Preparations
Point to Point Communication

Manual Version 4.1.2

Electrical and Information Technology

November 20, 2019

LUNDS UNIVERSITET
Lunds Tekniska Högskola

Table of Contents

Part I
Introduction

This lab is about the implementation of the necessary layer L1, L2, and L7 mechanisms to enable two units to communicate with each other over an Infra-red (IR) link. One unit, the *Master Node* will be provided for you and is fully functional according to the specifications in *Specifications* document. The *Development Node*, with which you will be working, comes with a complete set of Hardware (HW) and a Software (SW) skeleton companioned with and library of methods and variables. In this manual, you will find documentation of the communication standard, the *Master Node*, the HW for the *Development Node*, and a SW skeleton.

The reason why these two nodes need to communicate with each other is to be able to remotely set which Light Emitting Diode (LED) to illuminate on the other node. The *Master Node* is the node who's LEDs are controlled, while the *Development Node* acts as a remote control. Both nodes have three different-coloured LEDs and a button. While the button on the *Development Node* is pressed the LEDs light in sequence. When the button is released, the ID of the then lit LED shall be transmitted to the *Master Node*, where the same coloured LED shall be illuminated. Upon successful transmission the *Development Node* shall return to waiting for the next action by the operator. Similarly, after addressing the instructions in the received frame, the *Master Node* returns to waiting for the next instruction.

The nodes have limited memory, computational, and Input/Output (I/O) resources. This imposes constraints on the speed of communication, redundancy, and complexity of the application. The nodes are for example single threaded. It is therefore, non-trivial to run concurrent processes such as simultaneous transmission and reception on the device. These constraints have to be dealt with in your implementation.

This version of the lab manual is adapted to the skeleton version 5, `Skeleton5.ino`.

Table II.1: Predefined states (constants)

Name	Description
L1_PHY_RECEIVE	Rx: Receive frame on layer Layer 1 (L1)
L1_PHY_TRANSMIT	Tx: Transmit frame on layer L1
L2_LINK_FRAME_COMPOSE	Package the Layer 2 (L2) payload to be transmitted
L2_LINK_FRAME_DECOMPOSE	Process received payload on layer L2
L7_APP_PRODUCE	Produce content/message to send
HALT	"halt" the system, i.e. an infinite loop

Part II
Preporatory assignments

Read the lab instructions and review the Skeleton5.ino file before you begin with the assignments.

The two assignments below are compulsary. You are expected to present your answers to a TA at the lab session. Your answers to the below questions will be key to completing the lab.

Note that the state diagram in Figure II.1 is at the core of this lab. During the lab you will implement that state machine with the functions in Table II.2.

II.1 State machine

Figure II.1 is a representation of *Development Node*'s state machine. It is your task to complete the diagram by assigning the states from Table II.1 to the states in Figure II.1.

II.2 Function to state mapping

In Table II.3, using the functions from the librabry found in the specifications document, and in Table II.2 [1], specify which functions are neccecary to realise each state and the appropriate input values parameters. In the lab, you will are expected to implement the functainality of each state using the functions in Table II.2. It is therefore important that you understand what the is acmplished in each state and what each function does. Note that not all states have a corresponding set of functions.

[1] Note that the number of rows does not nessecarily reflect the number of functions.

Table II.2: Library functions

Module	**Function**	**Inputs**	**Outputs**
Payload	`Payload` (Constr.)	int data	Payload
Frame	`Frame` (Constr.)	Payload led, int src, int dst, int frame_type	Frame
Frame	`print`	n/a	void
Shield	`select_led`	n/a	int
Shield	`halt`	n/a	void
Shield	`get_address`	n/a	int
Transmitter	`transmit_frame`	Frame	void
Receiver	`receive_frame`	int timeout	RECEIVED

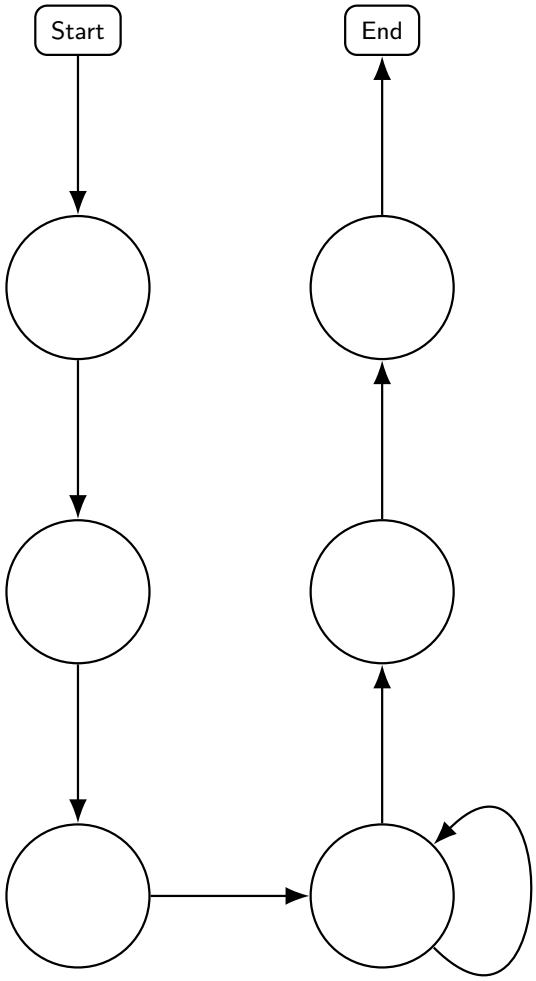

Figure II.1: *Development Node* states

Table II.3: Predefined states (constants)

State	**Module**	**Function**	**Inputs**	**Outputs**

Part III
Lab set-up

During the lab you will have access to a lab computer equipped with the Arduino Integrated Development Environment (IDE) [1] and a USB power supply. You will also have access to one *Master Node* and one *Development Node*. The *Master Node*, briefly described in Section III.4 and fully described in the *Specifications* document, will loop through the states depicted in *Specifications* document and you will not have access to manipulate or view its SW sketch. By default, the *Master Node* will be powered by the USB power supply. You can however connect it to the lab computer to view its debug output.

III.3 Practicalities

User account for logging in to the computers: telecomuser.

Open a terminal (system tools/Terminal) and run command arduino in the terminal, you should be able to open the Arduino IDE with a sketch file called Skeleton5. You can find this file created in the directory /home/Arduino/.

You can work with this sketch, but remember to save it with another name before you exit Arduino. Otherwise it will be overwritten if you run arduino command again in the terminal.

Please note that your sketch will also be deleted once you log out and the computer is restarted. Remember to save your own sketch to your own portable drive if you want to take it home. There no internet in the Sibirien lab, thus you can not save it to your network drive, neither can you access the documents online.

To set up the communication with the Arduino board ensure that

- `Tools–Board` is set to "Arduino/Genuino Uno".
- `Tools–Port` is set to "/dev/ttyACM0 (Arduino/Genuino Uno)". The port name can differ, but a similar text should be there.

There should be two Arduino boards on your working bench, one connected to the computer, i.e. the *Development Node*, marked with a D or T, and one connected to a USB power outlet, i.e. the *Master Node* marked with M or R. Try to upload the Skeleton sketch to the *Development Node*. This can be done using the icons in the upper left corner of the text editor.

For debugging purposes you can use either the three LEDs D3 to D5 on the shield (pins 7, 8 and 9) and use the `Serial` functions (see *Specifications* document for printing data to the Arduino IDE's *Serial Monitor*.

III.4 A Very Brief Introduction to the *Master Node*

The *Master Node* is a fully functioning remote host that can receive and decode a frame, and perform the remote host Automatic Repeat Request (ARQ) functionality. Optionally it can validate Cyclic Redundancy Check (CRC) parity bits of incoming frames as well create CRC parity bits for outgoing frames.

For this lab, on the *Master Node*, all four Dual In-line Package (DIP) switches should be set to 0.

References

[1] Arduino software (ide). https://www.arduino.cc/en/Guide/Environment, 2015.