# Lab 2: Receiving information

April 8, 2025

## 1 Learning outcomes

After performing lab 2, you should have gained understanding of

- how to receive a frame by sampling a signal, detect symbols, synchronise (i.e. finding Start Frame Delimiter (SFD)), and decoding the received symbol train into frame fields bits.

## 2 Lab Tasks

Lab one and two is about the implementation of the necessary Physical Layer (L1) mechanisms to enable two units to communicate with each other over an Infra-red (IR) link. In this lab you have to:

- Implement the necessary functionality in the *Development Node* so that it is able to receive frames from the static and predictable *Master Node*.

## 3 Task 1

In task 1 we start looking into receiving. When receiving pulses, you have to synchronise to the sending device, in this case the *Master Node*. The first thing is to wait for the reception of the preamble. When the start of the preamble is detected, you have to start sampling the pulses, convert pulses to bits and store these bits in a receive buffer. Try to find out what type of frame you have received.

### 3.1 Hints

- At what layer are pulses first received - which state should this task therefore be implemented in?

- As in task 2, it is suggested to write the 'receiver' as a function, e.g. `boolean l1_receive(int timeout)` which is called with e.g. `recOK = l1_receive(20000)`. It might not be the best practise, but it is an easy solution to allow the receiver function to store bits in a globally declared object.
- Remind yourself on bitwise operations by reading in the *Reference Manual*, the operations and examples described here will be useful here as well.
- Write code for the reception of the frame that the *Master Node* sends as a response to the frame you sent to it. The receiver should continuously listen to the channel until the front of the frame, i.e. a preamble pulse equal to a binary 1, is detected. Use the method `Shield::sampleRecCh(PIN_RX)`.
- Then the receiver should sample all pulses into bits, i.e. sample pulses on `PIN_RX` with sample time $T_s$ and convert the pulses to bits. Each bit should then be left shifted in to a byte sized buffer, the content of which is to be compared with the SFD in the next task. Print out the content of this buffer on the *Serial Monitor* for each sampled symbol. Manually find the SFD in the *Serial Monitor* print-out. Time for execution of code between samples have to be taken into consideration. How will your code adapt to this fact?
- A time-out should be implemented. If the preamble is found within the time-out, the receiver should stop sampling after a full frame is received and return a message of success. If not, the receiver should return a message of fail. Also, let debug Light Emitting Diode (LED) 1, `DEB_1`, follow the sampled symbols when receiving.

## 4 Task 2

In the last task, you should decompose the received frame into Data Link Layer (L2) variables. For this you have to find where the L2 frame starts in the train of received bits. This task is done when you can correctly print out the individual frame fields on the serial monitor.

### 4.1 Hints

- In this task you have to add finding the SFD to the receiver. You have to find the end of the SFD, which marks the start of the L2 frame. This is discussed in the lectures and in the *Reference Manual*. In our case the SFD is the size of one byte, and we can use this to our convenience by using a byte as the input buffer as in the previous task and test it against the global constant for the SFD.
- Once the SFD has been detected, turn on the first debug LED `DEB_1` and let the second debug LED `DEB_2` follow the value of the sampled symbols. Light up the LED `DEB_2` when the full frame is received.

- After finding the SFD, sample all the bits of the L2 frame into the L1–L2 interface i.e. the receive frame. As an alternative, you can store the sapled bits in a receive buffer, and when the full frame is received copy this buffer to the L1–L2 interface.
- Note that the receiver time-out from the previous task now has to include also the added SFD function.
- Finally, you need to decompose the received L2 frame into variables (in which state should this be done?). You can do the decomposition of the received frame in a corresponding way as you did the composing of the frame that was transmitted . Search the *Reference Manual* for a suitable method.
- Try to change the behaviour from going to a halt after each send–receive pair, to continuously loop send–receive–send–receive– and so on.
- Finally, store your produced sketch.