Fiber-Optic Communication System

Florida International University
Department of Electrical and Computer Engineering

Nestor Viana, Gustavo Ramirez

April 2024

Abstract

We create a communication channel that transmits and receives data via Amplitude Shift Key (ASK) modulation of electromagnetic carrier waves. In particular, we will adjust a potentiometer to a resistance value of R and this value will be transmitted via fiber optic cable and displayed on an LCD screen connected to the receiver part of the system.

1 Introduction

A digital communication system is a communication method that uses digital sequences where the source of information output is converted into binary sequences to be transmitted over a physical medium such as wire, sound, light, etc., to a destination channel where the data is decoded through a suitable interface. This method has become today's standard for communication thanks its reliability, scalability, and inexpensive manufacturing of its components. In contrast to traditional copper wire communication, fiber optic communication channels enable higher speeds, bandwidth, and distances with low interference effects [2]. This is crucial for modern applications like high-definition video streaming, cloud computing, and real-time data transfer. In fiber optic communication, Amplitude Shift-Key (ASK) modulation is particularly advantageous due to its simplicity and compatibility with optical fibers. In this project, we demonstrate how to build a simple fiber optic communication channel that implements an ASK modulation scheme.

1.1 ASK Theory

ASK modulation is a digital modulation technique commonly used in fiber optic communication to encode digital data onto an optical carrier signal. The amplitude of this carrier signal is modulated in accordance to the type of digital data transmitted, typically binary 1s and 0s. For example, sending the bit stream 01001110 modeled by $x(t)$, with carrier signal $\sin(\omega t)$, produces the modulated signal

$$y(t) = x(t)\sin(\omega t).$$

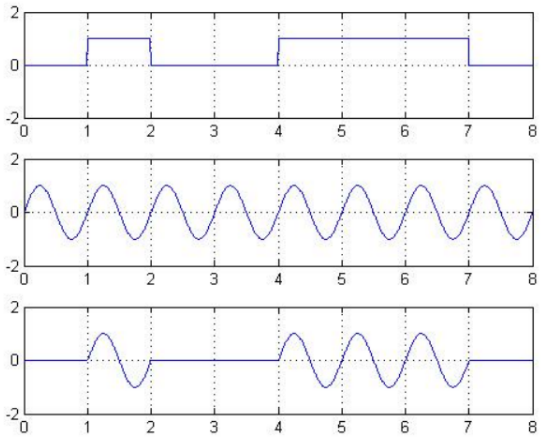

Figure 1: ASK modulation example; from top to bottom: Data signal $x(t)$, carrier wave, modulated carrier wave $y(t)$.

2 Methodology

The first step in optical fiber communication is to convert the electrical signals into optical signals. This can

be done by sending a signal down an LED or laser diode or through a modulator that uses an electrical signal to control the intensity of light emitted from a light emitting diode once we have converted electrical signals into optical ones. These optical signals then travel through a fiber optic channel until they reach the receiver. The receiver module then converts the optical signals back into electrical signals which can be decoded, typically together with an error-correction scheme [1].

We use a HFBR 1414 fiber optic transmitter module produced by Broadcom. It is a low-cost and high-power transmitter that can send data at rates up to 160 Mbps over distances of up to 2.7 km and implements an 820 nm emitter module. This transmitter is able to operate at low current inputs which results in lower power consumption. Additionally, it is optimized for small-scale optical fiber sizes with power ranging from -15.8 dBm at 60 mA into 50/125 μm fiber and -12 dBm into 62.5/125 μm fiber. For the receiver module, we utilize a HFBR 2412 receiver which is compatible with the HFBR 1414 to complete the fiber optical communication link, capable of receiving information at a rate of up to 5 Mbps over a distance of 2 km. Finally, we use a multimode fiber optic cable with core diameter of 62.5 μm, and the cladding diameter 125 μm.

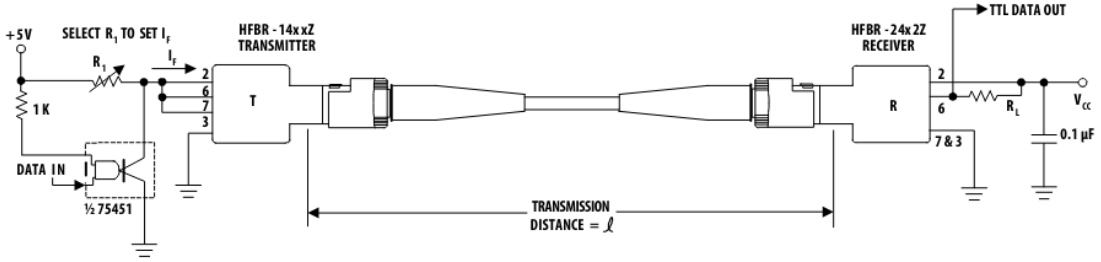

Figure 2: Transmitter-Receiver module schematic [3].

Both transmitter and receiver modules will be connected to Arduinos to properly program the functionality of each system. The Arduino on the transmitter system will read the value of R_1 (on Figure 2), and convert this data using the RH_ASK.h library on C to electrical signals that will modulate the 820 nm carrier wave. Similarly, the RH_ASK.h library will be implemented to program the Arduino on the receiver side to decode the optical pulses into electrical signals, and finally display the value of R_1 on an LCD screen.

3 Results

Using the schematic on Figure 2, we connect all the modules to the corresponding Arduinos. The code used for each Arduino are be provided at the end of this report.

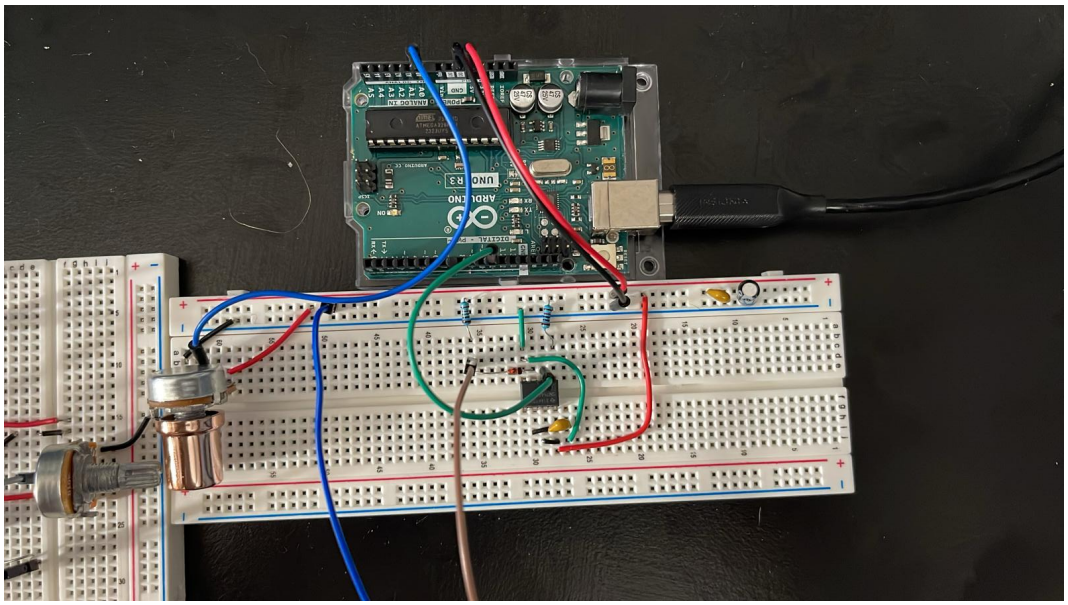

Figure 3: Transmitter system.

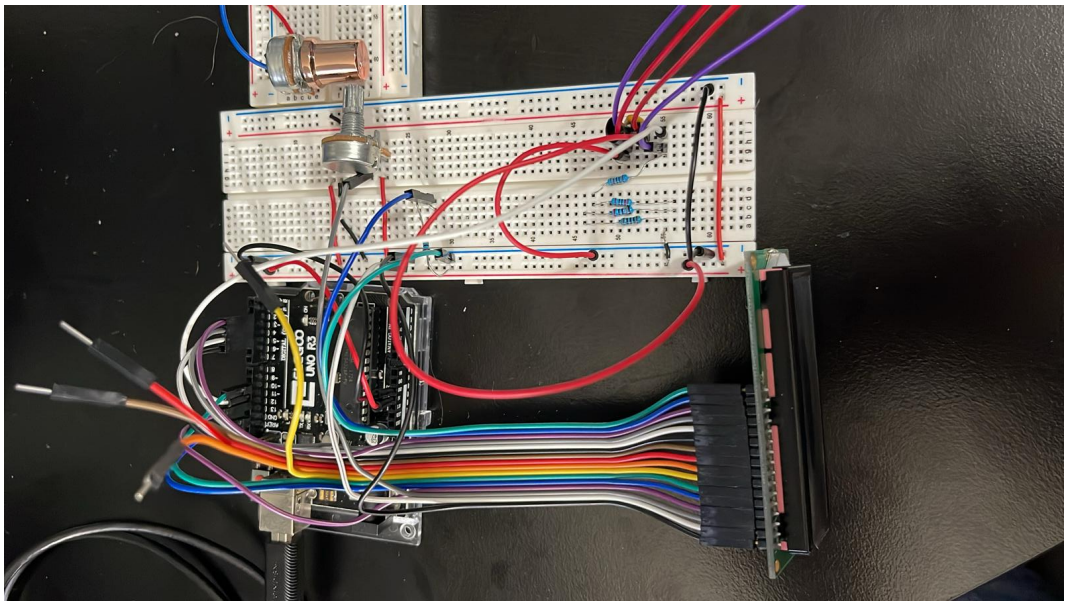

Figure 4: Receiver system.

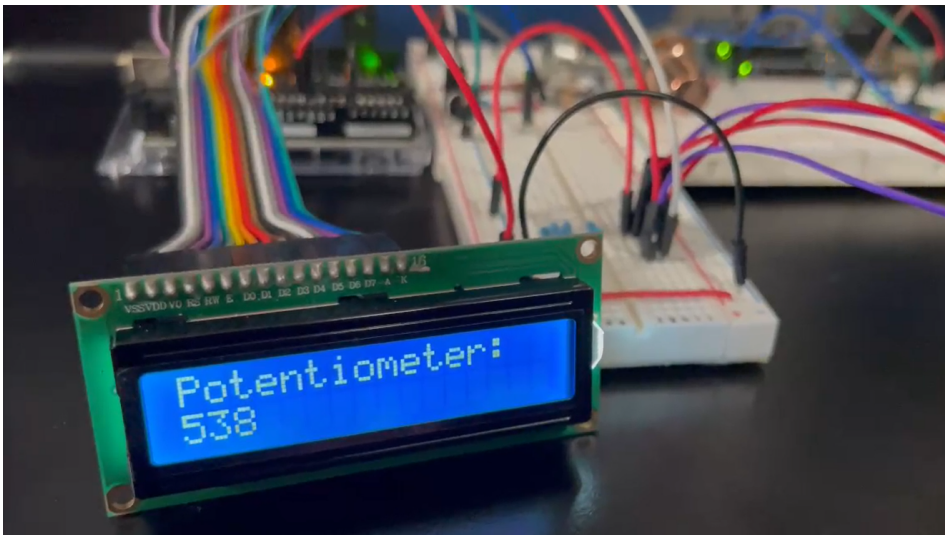

Figure 5: LCD screen with potentiometer value of R_1.

References

[1] In: *Wavelength Electronics* (Dec. 2022). URL: https://www.teamwavelength.com/modulation-basics/.

[2] S Babani et al. "Comparative study between fiber optic and copper in communication link". In: *Int. J. Tech. Res. Appl* 2.2 (2014), pp. 59–63.

[3] *HFBR-14xxZ and HFBR-24xxZ Series: Data Sheet*. https://docs.broadcom.com/doc/AV02-0176EN. (Accessed on 04/27/2024).

Receiver module Arduino code:

```
#include <LiquidCrystal.h>
#include <RH_ASK.h>

RH_ASK driver;
int potValue;
char receivedData[5];

const int rs = 12, en = 10, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  if (!driver.init())
    Serial.println("RadioHead initialization failed");
    driver.available();
}

void loop() {
  uint8_t buf[RH_ASK_MAX_MESSAGE_LEN];
  uint8_t buflen = sizeof(buf);

  if (driver.recv(buf, &buflen)) {
    if (buflen < 5) {
      for (int i = 0; i < buflen; i++) {
        receivedData[i] = (char)buf[i];
      }
      receivedData[buflen] = '\0';

      potValue = atoi(receivedData);

      lcd.print("Potentiometer: " + \n potValue);
    }
  }
}
```

Transmitter module Arduino code:

```
#include <RH_ASK.h>

RH_ASK driver;

int potPin = A0;
int potValue;

void setup() {
  Serial.begin(9600);
  if (!driver.init())
    Serial.println("Initialization failed");
}

void loop() {
  potValue = analogRead(potPin);

  char dataToSend[5];
  itoa(potValue, dataToSend, 10);

  driver.send((uint8_t *)dataToSend, strlen(dataToSend));
  driver.waitPacketSent();

  Serial.println(potValue);

  delay(100);
}
```