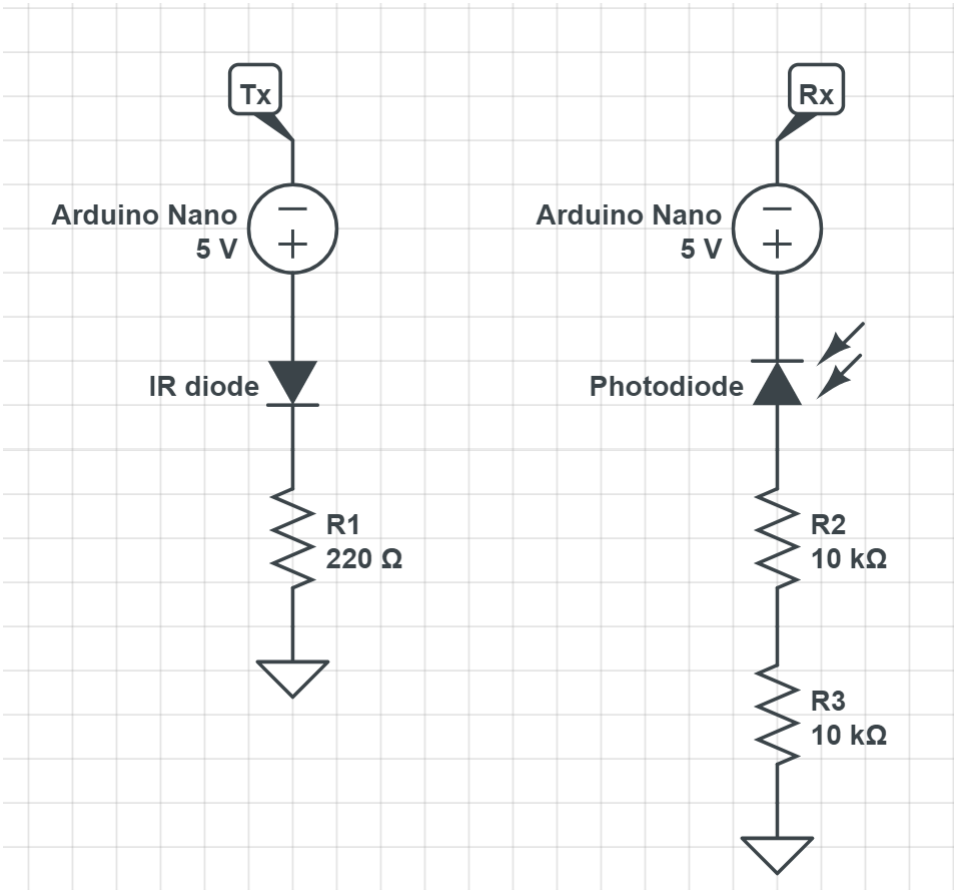
IR Photodiode Transmitter – Arduino Project

By Yuval Horowitz & Ariel Turnowski

1. Background on the project

As part of this project, we were asked to produce an infrared-based communication line between two Arduino cards using a diode and a photodiode functioning as the output and input respectively, using simple communication principles expressed in the "UART" protocol.



2. Work steps
   a) Preparation - learning the technical specifications of the Arduino, using the electronic components and the matrix, understanding the programming language and the programming environment.
   b) Understanding the diode and photodiode component and building the circuits.
   c) Reading the state of the photodiode - experimenting with reading input from the electronic component and getting a result in the software environment.
   d) Transferring information between the Arduinos - experimenting with transferring information between diodes and reading it using the photodiode.
   e) Writing software that translates a string into a binary string, passes them in sync between the transmitter and the receiver and decodes it on the second Arduino.
3. Hardware

a) The output circuit (infrared LED)

- The components of the circuit - infrared LED, resistor (220 Ohm), Arduino chip.
- The Arduino provides a voltage of 5 volts, so we calculated that to pass a current of 15 milliamps through the LED we would need a resistor of 220 Ohms, according to Ohm's law I*R) = V).

b) input circuit (photodiode)

- The circuit components - photodiode, 2X resistor (10K ohm), Arduino chip.
- The Arduino provides a constant voltage of 5 volts to the photodiode, which is connected on the other side to the two resistors connected to the Arduino leg. When the component receives light, it sends a current that is translated through the resistors into a voltage that the Arduino knows how to receive. We noticed during the experiment that a high resistance is needed in order for the Arduino to be able to detect the signal (since the current is very low).

4. <u>Software</u>

The transmission algorithm we chose for communication between the components is a combination of the "threshold" principle and "UART" communication.
The communication between the component and the user was carried out through a serial-monitor channel that allows this.

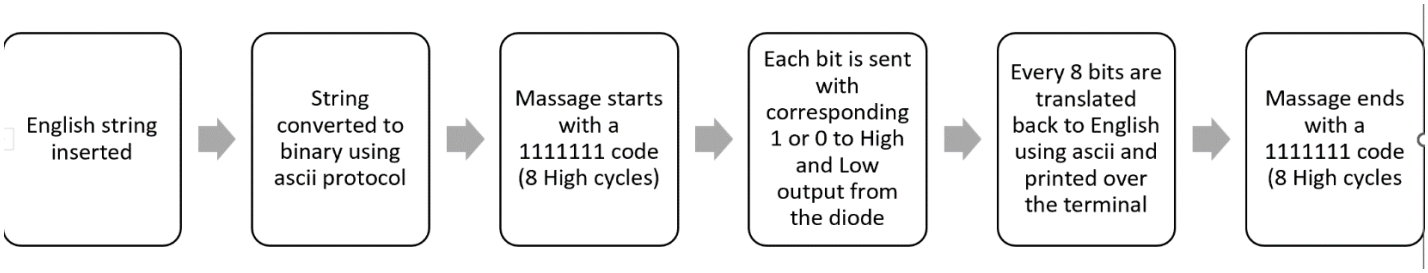
a) The transmission algorithm

- The "threshold" principle comes to me in the following way, the component transmits a signal for a fixed time that varies according to the signal we want to transmit. For '1' it is broadcast for a seconds, and for '0' it is broadcast for b seconds. In the algorithm we determined a threshold time whose size is between a and b and this is the sampling time of the input device.
- The transmission code has two central functions: definition- defines the system, translation- translates an English string into a binary string by using ascii encoding, and a loop which transmits the signals.
    - Definition - in this function we defined the transmission leg of the Arduino component so that it knew how to supply voltage according to the requirement in the code. In addition, we defined the interface (serial-monitor) through which we determine the string we want to transmit.
    - Loop and transmission - the function transmits a high voltage for 8 cycles (Byte) and starts transmitting the binary string when it transmits a high voltage when the value is 1 and a low voltage when the value is 0, at the end of the string the function transmits a high voltage for 8 cycles (Byte)

b) Absorption algorithm

- The photodiode is connected to one of the legs of the Arduino so that if a voltage is received in the leg we identify it in the software as an analog value.
- The algorithm reads signals continuously and looks for a string of 8 high cycles (Byte) then rewrites the sequence it receives and each byte converts the binary string to a signal in the message and prints it, this is repeated until a string of 8 high cycles is received and the coding stops.
- The reception code has two central functions: the reception loop definition and translation.
    - Definition - in this function, the interface is initialized to display the input and define the receiving leg from the component. Also configure the internal light on the Arduino so that it lights up when a signal is received.
    - The translation loop = receives an 8-bit binary string and returns the corresponding value according to ascii encoding.
    - The recording loop - there are several while loops inside the loop, the first is interrupted when 8 high cycles are received, the second is interrupted every 8 bits received or when a break string is received (8 high cycles). After "synchronizing" using the first string, the loop prints the character corresponding to the received binary string until a string of 8 high bits is obtained.

5. <u>The process of reception and transmission</u>



6. <u>Summary and conclusions</u>

a) The communication project in IR contained a combination of hardware and software in a seemingly basic way, but it turned out to be composed of large and integrated principles of hardware-software interface and communication.

b) The best bit rate we have reached is - 50 milliseconds per bit without errors.

Appendices

Reciever code

```
int ReadPin =A3;
float V=0;
float fix = 5./1023.;
float readVal = 0;
String msg = "";
String F ="";
int finish_counter = 0;
int delaytime = 50;
bool end = false;
bool newmsg = false;
void setup()
{
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(ReadPin,INPUT);
  //ascii to english part:
}
String asciiBinaryToString(const String& asciiBinary) { //Transalate an 8 bit binary string to a char
  String result = "";

  // Ensure that the length is a multiple of 8 for valid ASCII binary
  int length = asciiBinary.length();
  int padding = length % 8;
  if (padding != 0) {
    length += (8 - padding);
  }

  // Process each group of 8 bits
  for (int i = 0; i < length; i += 8) {
    char currentChar = 0;

    // Convert binary to character
    for (int j = 0; j < 8; ++j) {
      if (i + j < asciiBinary.length()) {
        currentChar <<= 1;
        currentChar |= (asciiBinary[i + j] == '1') ? 1 : 0;
      }
    }

    // Append the character to the result
    result += currentChar;
  }
  return result;
}

void loop() {
  while (newmsg == false){ // waits to get a 1 byte to start reading a msg
  V = analogRead(ReadPin) * fix;
  //Serial.println(V);
  delay(delaytime);
  if(V>1.00){
    msg.concat("1");
  }
  else {
    msg.concat("0");
    msg = "";
    }
  if (msg == "11111111"){
    msg = "";
    newmsg = true;
    Serial.println("New string incoming!");
    }
    }
  while((end == false) &&(finish_counter!=8 && msg.length()%8 !=0) || msg.length()==0) //read each bit of the msg
  {
    readVal  = analogRead(ReadPin);
```

```
    V = readVal*fix;
    if (V>1.0)
    {
      msg.concat("1");
      //Serial.println(msg);
      finish_counter +=1;// count if you reach the end of the msg
      delay(delaytime);

    }
    else
    {
      msg.concat("0");
      //Serial.println(msg);
      finish_counter = 0;
      delay(delaytime);
    }
  }
  if (finish_counter !=8) // when having an entire byte translate it to englis
  {
    Serial.print(asciiBinaryToString(msg));
    finish_counter=0;
    msg="";
  }
  else
  {
    end = true;
  }
  }
```

Emitter Code

```
int readPin = A3;
float V = 0;
float fix = 5./1023.;
float readVal = 0;
int delayTime=50;
//String binaryResult="0110100001100100";

int letters_num = 0 ;
bool done = false;

String stringToBinaryASCII(String inputString) {
  String binaryRepresentation = "";

  for (int i = 0; i < inputString.length(); i++) {
    char currentChar = inputString.charAt(i);
    byte asciiValue = (byte)currentChar;

    for (int j = 7; j >= 0; j--) {
      binaryRepresentation += (char)('0' + ((asciiValue >> j) & 1));
    }
  }

  return binaryRepresentation;
}
String myString = "Davidc";
String binaryResult = stringToBinaryASCII(myString);
void setup() {
  // put your setup code here, to run once:
  pinMode(10,OUTPUT);
  Serial.begin(9600);
  pinMode(readPin,INPUT);
}

```

```
void loop() {

    Serial.println(binaryResult);
    for (int i =0; i <8;i++){
    digitalWrite(10,HIGH);
    readVal  = analogRead(readPin);
    V = readVal*fix;
    Serial.println(V);
    delay(delayTime);
    }
    for (int i =0;i<binaryResult.length();i++)
    {
        char currentch = binaryResult[i];
        if (binaryResult[i] == '1')
        {
          digitalWrite(10,HIGH);
            readVal  = analogRead(readPin);
            V = readVal*fix;
            Serial.println(V);
            delay(delayTime);
        }
        else
          {
            digitalWrite(10,LOW);
            readVal  = analogRead(readPin);
            V = readVal*fix;
            Serial.println(V);
            delay(delayTime);
          }
    }
    for (int i =0; i <8;i++){
      digitalWrite(10,HIGH);
      readVal  = analogRead(readPin);
      V = readVal*fix;
      Serial.print(V);
      delay(delayTime);
    }
```

Hardware Layout