Московский физико-технический институт
Физтех-школа радиотехники и компьютерных технологий

(Микроконтроллеры)

Кодирование и декодирование азбукой Морзе

Работу выполнили:
Зайцев Василий, Б01-904
Костенок Елизавета, Б01-904
Мирзоян Мери, Б01-003

г. Долгопрудный
2021 год

1 Цель:

Собрать и запрограммировать систему, осуществляющую кодирование текстовой строки азбукой Морзе и декодирование обратно в текстовую строку.

2 Описание проекта:

Система состоит из 3 частей: передатчик, канал связи, приемник.

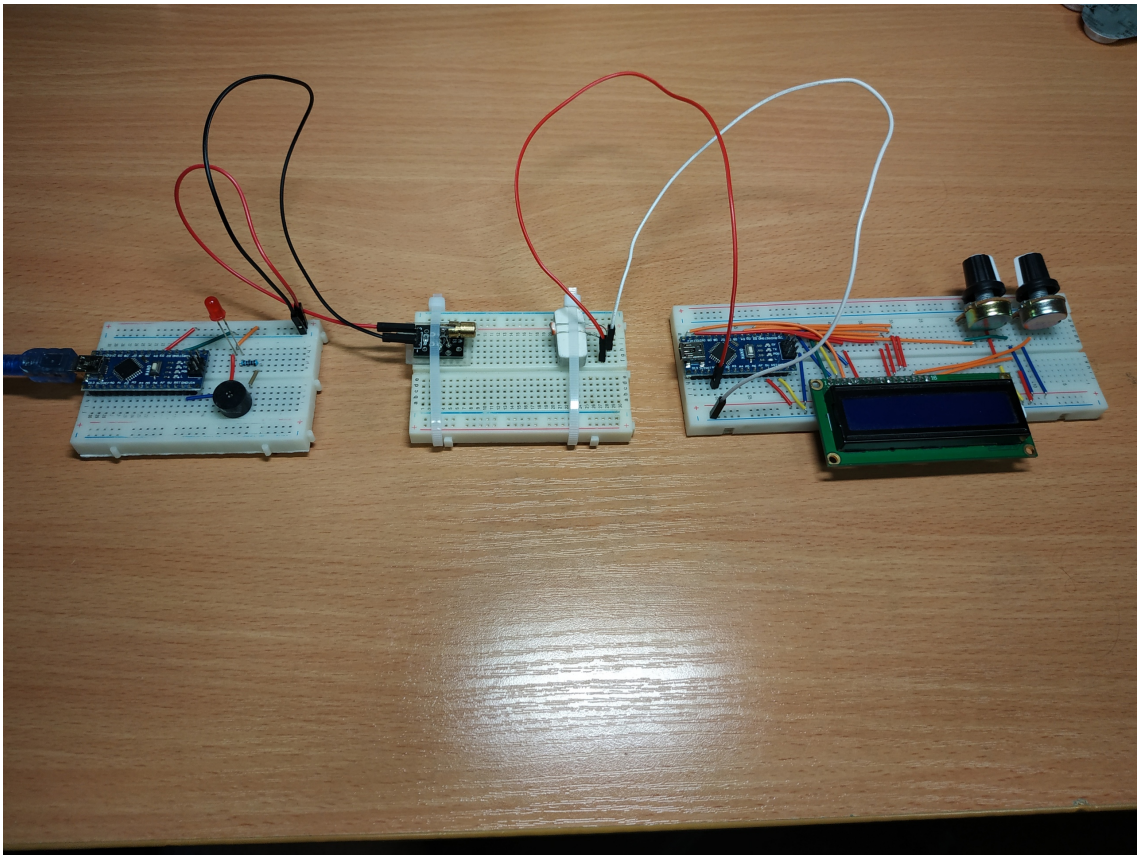
- Передатчик представляет собой макетную плату с платой arduino nano на МК Atmega328p, которая преобразует текстовую информацию в цифровой сигнал (точки и тире, соответствующие азбуке Морзе), он передается на "пищалку" , светодиод (для визуальной индикации передачи), а так же на лазерную головку, которая является началом канала связи.

- Канал связи: оптический, представляет собой лазерный луч в воздухе. Канал связи помещен в непрозрачную коробку для лучшей помехоустойчивости. Для передачи на большие расстояния данный канал связи может быть заменен на оптоволокно.

- Приемник: Лазерный луч воспринимается фоторезистором, который изменяет свое сопротивление в зависимости от освещенности, таким образом, мы получаем аналоговый сигнал, по которому МК определяет отсутствие или наличие точек и тире в сообщении. Полученную информацию он декодирует обратно в текстовую информацию, которая затем поступает на LCD-дисплей. 2 потенциометра на плате нужны для того, чтобы менять подсветку и контрастность дисплея

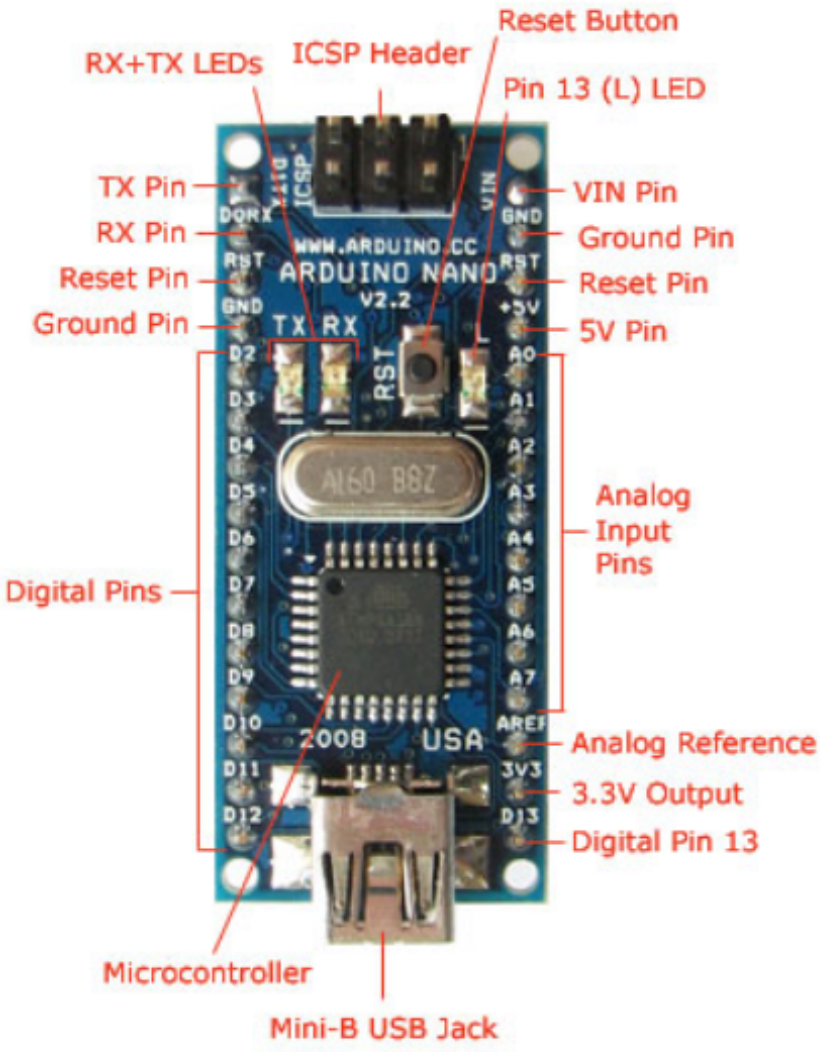
3 Схема установки

2 микроконтроллера Atmega328p на платах Arduino nano, lcd-дисплей, светодиод, лазер, фоторезистор, провода, потенциометры.



4 Платформа Arduino Nano

- Микроконтроллер Atmel ATmega168 или ATmega328
- Рабочее напряжение (логический уровень) 5 В
- Входное напряжение (рекомендуемое) 7-12 В
- Входное напряжение (предельное) 6-20 В
- 14 цифровых входов/выходов (6 из которых могут использоваться как выходы ШИМ)
- 8 аналоговых входов
- Флеш-память 16 Кб (ATmega168) или 32 Кб (ATmega328) при этом 2 Кб используются для загрузчика
- ОЗУ 1 Кб (ATmega168) или 2 Кб (ATmega328)
- EEPROM 512 байт (ATmega168) или 1 Кб (ATmega328)
- Тактовая частота 16 МГц



5 Микроконтроллер ATmega328P:

ATmega328 - однокристальный микроконтроллер, созданный Atmel в семействе megaAVR. Он имеет модифицированное ядро 8-битного RISC-процессора с гарвардской архитектурой, высокоскоростные встроенные запоминающие устройства и широкий спектр расширенных периферийных устройств ввода-вывода.

- High performance, low power AVR 8-bit microcontroller
- Advanced RISC architecture: 131 powerful instructions – most single clock

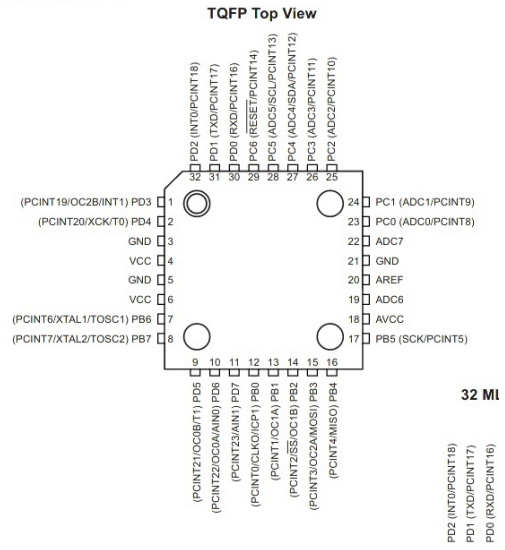
cycle execution, 32 * 8 general purpose working registers, fully static operation, up to 16MIPS throughput at 16MHz, on-chip 2-cycle multiplier

- High endurance non-volatile memory segments: 32K bytes of in-system self-programmable flash program memory, 1Kbytes EEPROM, 2Kbytes internal SRAM, write/erase cycles: 10,000 flash/100,000 EEPROM, optional boot code section with independent lock bits, in-system programming by on-chip boot program, true read-while-write operation, programming lock for software security

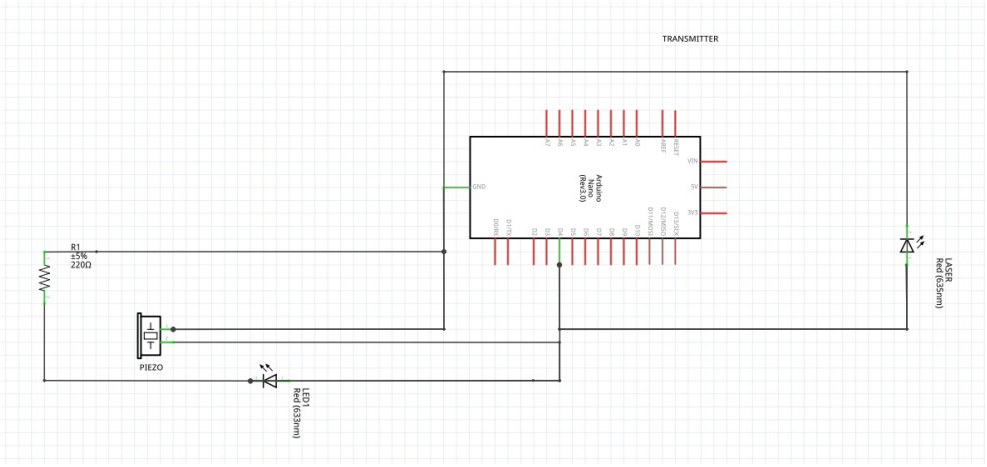
- Peripheral features: two 8-bit Timer/Counters with separate prescaler and compare mode, one 16-bit Timer/Counter with separate prescaler, compare mode, and capture mode, real time counter with separate oscillator, rix PWM channels, 8-channel 10-bit ADC in TQFP and QFN/MLF package, temperature measurement, trogrammable serial USART, master/slave SPI serial interface, byte-oriented 2-wire serial interface (Phillips I2 C compatible), programmable watchdog timer with separate on-chip oscillator, on-chip analog comparator, interrupt and wake-up on pin change

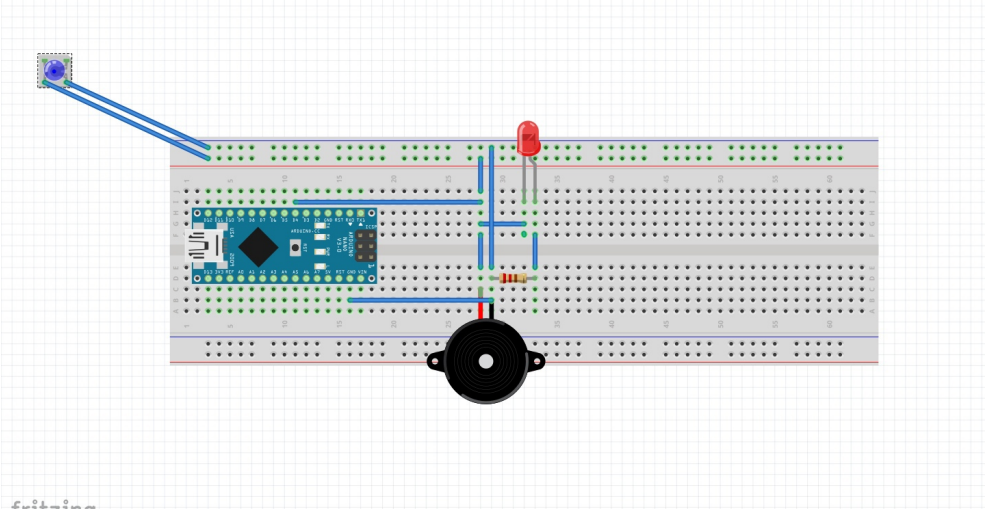
- Special microcontroller features:
- Power-on reset and programmable brown-out detection
- Internal calibrated oscillator
- External and internal interrupt sources
- Six sleep modes: Idle, ADC noise reduction, power-save, power-down, standby, and extended standby
- Operating voltage: 2.7V to 5.5V for ATmega328P
- Speed grade:
- 0 to 8MHz at 2.7 to 5.5V (automotive temperature range: –40°C to +125°C)
- 0 to 16MHz at 4.5 to 5.5V (automotive temperature range: –40°C to +125°C)
- Low power consumption
- Active mode: 1.5mA at 3V - 4MHz
- Power-down mode: 1µA at 3V

Figure 1-1. Pinout



6 Transmitter:

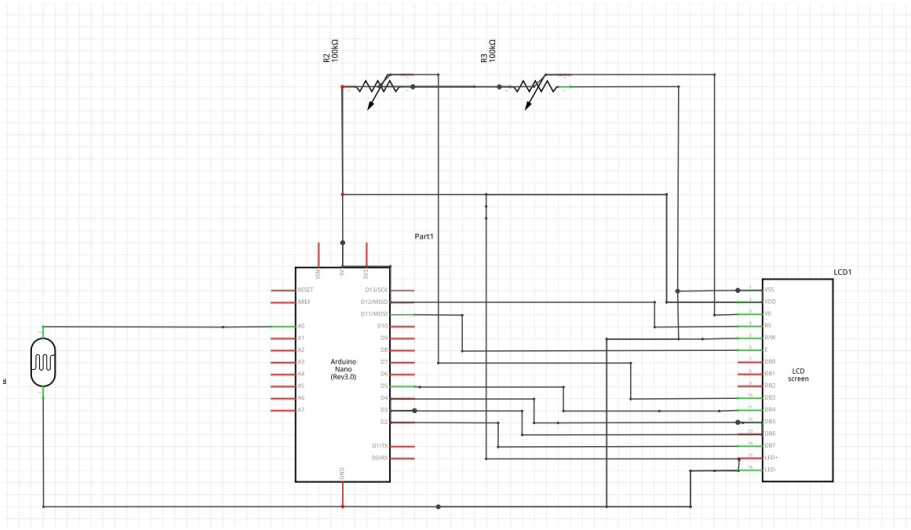


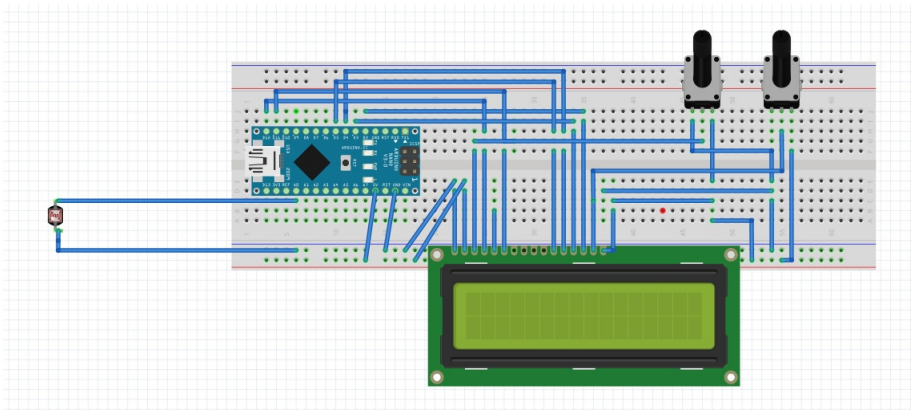


```
const uint32_t DOTTIME = 100;
const uint8_t SIGNALPIN = 4;

void LexFlash(const char* letter);
void Flash(int n);

//Morse codes (0 is DIT, 1 is DAH)
const char* letters[] =
{
  "01", "1000", "1010", "100", "0", "0010", "110", "0000", "00", //A to I
  "0111", "101", "0100", "11", "10", "111", "0110", "1101", "010", //J to R
  "000", "1", "001", "0001", "011", "1001", "1011", "1100"         //S to Z
};

const char* numbers[] =
{
  "11111", "01111", "00111", "00011", "00001", //0 to 4
  "00000", "10000", "11000", "11100", "11110"  //5 to 9
};

const char* spec_symb[] =
{
  "000000", "101010", "010101", "111000",     //". ; , :"
  "001100", "110011", "100001"                //"? ! -"
};

void setup()
{
  pinMode(SIGNALPIN, OUTPUT);
  Serial.begin(9600);
}
```

```
void loop()
{
  static char lex = 0;  //1 letter from Serial
  if (Serial.available() > 0)
    lex = Serial.read();

  if ((lex >= 'a') && (lex <= 'z'))
    LexFlash(letters[lex - 'a']);
  if ((lex >= 'A') && (lex <= 'Z'))
    LexFlash(letters[lex - 'A']);
  if ((lex >= '0') && (lex <= '9'))
    LexFlash(numbers[lex - '0']);

  switch (lex)
  {
    case '.' : { LexFlash(spec_symb[0]); break; }
    case ';' : { LexFlash(spec_symb[1]); break; }
    case ',' : { LexFlash(spec_symb[2]); break; }
    case ':' : { LexFlash(spec_symb[3]); break; }
    case '?' : { LexFlash(spec_symb[4]); break; }
    case '!' : { LexFlash(spec_symb[5]); break; }
    case '-' : { LexFlash(spec_symb[6]); break; }
  }

  if (lex == ' ')
    delay(DOTTIME * 7);
}



void LexFlash(const char* letter)
{
  int i = 0;
  while (letter[i] != '\0')
  {
    if (letter[i] == '0')
      Flash(1);
    if (letter[i] == '1')
      Flash(3);
    delay(DOTTIME);   //Space between dits & dahs
    i++;
  }
  delay(DOTTIME * 3); //Space between letters
}

void Flash(int n)
{
  digitalWrite(SIGNALPIN, HIGH);
  delay(DOTTIME * n);
  digitalWrite(SIGNALPIN, LOW);

}
```

7 Receiver:

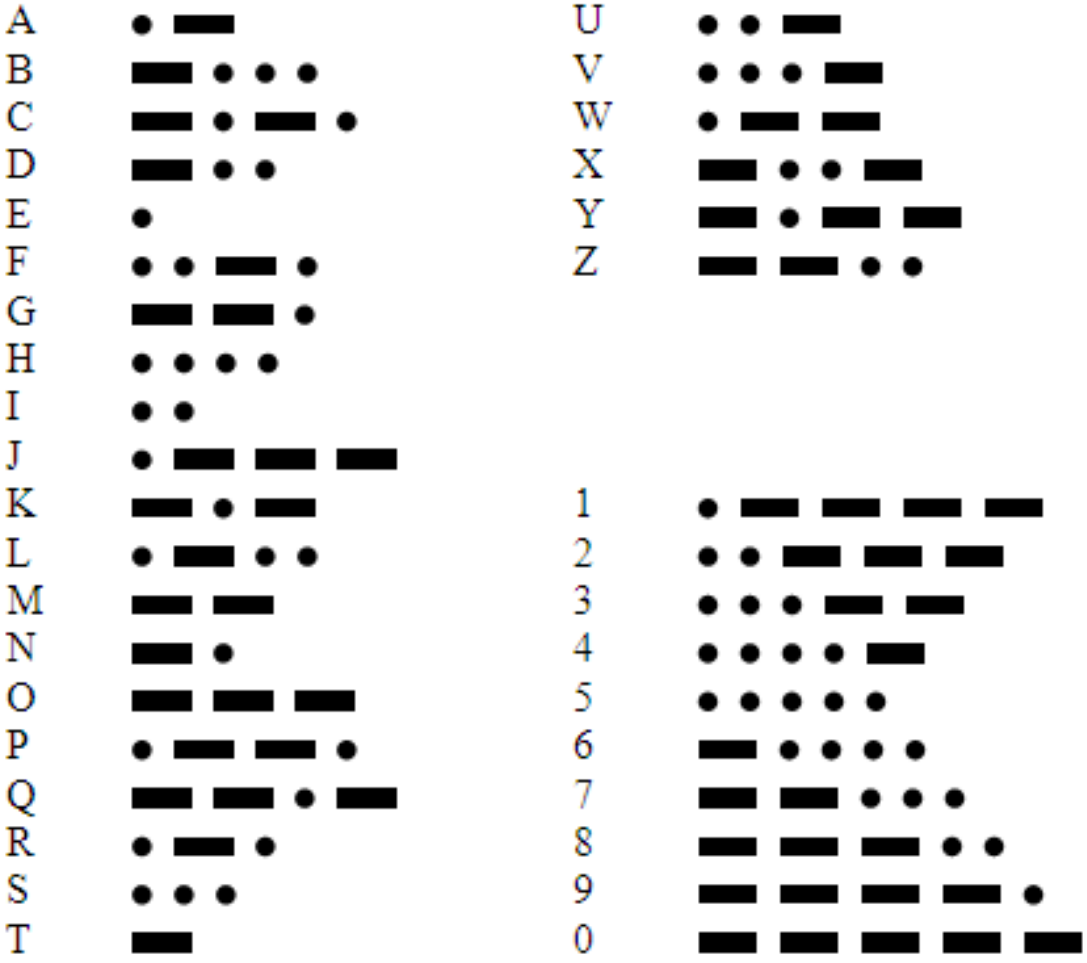




```
#include <LiquidCrystal.h>

/* This program is a Morse-code decoder

   Zaitsev Vasilii
   Elizaveta Kostenok
   Mary Mirzoian
   MIPT 2021
 */

/* NOMINAL CONSTANTS:
 *
 * DOT_TIME = 100 (1 unit)
 * DAH_TIME = 300 (3 units)
 * SPACE_BETWEEN_SYMBOLS = 100 (1 unit)
 * SPACE_BETWEEN_LETTERS = 400 (4 units)
 * SPACE_BETWEEN_WORDS = 700   (7 unit)
 */

//Constants
enum dot {DOT = 0, DAH = 1};
const uint8_t  pinLaser = A0;          //Pin with photosensor on Arduino
const uint8_t  waitDelay = 20;         //Spending time on pause/signal
const uint16_t laserLimit = 200;       //Analog difference between high & low signal
const uint16_t dotUnit = 200;          //Duration of one DOT unit
const uint32_t spaceUnit = 600;        //Duration of space between words
const uint32_t terminateUnit = 1000;   //Duration of pause "end of transmittion"

char      chrMorze[] =
{
 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T',
 'U', 'V', 'W', 'X', 'Y', 'Z',
 '1', '2', '3', '4', '5',  '6',  '7',  '8',  '9',  '0',
  '.',  ';',  ',',  ':',  '?',  '!',  '-'
};

//Morse codes (1 is DAH, 0 is DIT)
uint8_t   varMorze[] =
{
  0x1, 0x8, 0xA, 0x4, 0x0, 0x2, 0x6, 0x0, 0x0, 0x7,
  0x5, 0x4, 0x3, 0x2, 0x7, 0x6, 0xD, 0x2, 0x0, 0x1,
  0x1, 0x1, 0x3, 0x9, 0xB, 0xC, 0xF, 0x7, 0x3, 0x1,
  0x0, 0x10, 0x18, 0x1C, 0x1E, 0x1F, 0x0, 0x2A, 0x15,
  0x38, 0x0C, 0x33, 0x21
};




//Morse code lengthes
uint8_t   lenMorze[] =
{
  2,   4,   4,   3,   1,   4,   3,   4,   2,   4,
  3,   4,   2,   2,   3,   4,   4,   3,   3,   1,
  3,   4,   3,   4,   4,   4,   5,   5,   5,   5,
  5,    5,    5,    5,    5,    5,   6,    6,  6,
  6,    6,    6,    6
};

//Variables
bool flagSignal = false;      //Signal recieving now
bool flagLetter = false;      //Letter recieved now
bool flagSpace = false;       //Space between words recieved
uint8_t currSymbol = DOT;     //Current symbol
uint8_t currLetter = 0;       //Current letter
uint8_t lenLetter = 0;        //Length of the current letter
uint32_t tmp_terminate = 0;   //variables for counting time between events
uint32_t tmp_separator = 0;
uint32_t tmp_space = 0;


LiquidCrystal lcd ( 12 , 11 , 5 , 4 , 3 , 2 );


void setup ()
{
  //LCD setup
  lcd.begin ( 16 , 2 );        // specify the type of display LCD 16X2
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print ( "Morse code");
  lcd.setCursor(0,1);
  lcd.print ( "Decoder");
  delay(1000);
  lcd.clear();
  lcd.noAutoscroll();

  //Pin & serial port setup
  pinMode(pinLaser, INPUT_PULLUP);
  Serial.begin(9600);
}
```

```
void loop ()
{
  //If signal detected:
  if (analogRead(pinLaser) < laserLimit)
  {
    flagSignal = true;
    tmp_separator = millis() + dotUnit;     //To distingush, if it's dot or dah
    while(analogRead(pinLaser) < laserLimit) {delay(waitDelay);};
    if (millis() > tmp_separator)
      currSymbol = DAH;
    else
      currSymbol = DOT;

    currLetter <<= 1;
    currLetter |= currSymbol;
    lenLetter ++;
  }
  //If pause detected
  else
  {
    tmp_terminate = millis() + terminateUnit; //To distingush, if it's end of transmittion
    tmp_space = millis() + spaceUnit;         //To distingush, if it's space between words
    tmp_separator = millis() + dotUnit;       //To distingush, if it's space between letters
    while(analogRead(pinLaser) > laserLimit && (millis() < tmp_terminate)) { delay(waitDelay); }
    if (millis() > tmp_terminate)
    {
      flagSpace = true;
      flagLetter = flagSignal;
      flagSignal = false;

    }
    else if (millis() > tmp_space)
    {
      flagSpace = true;
      flagLetter = flagSignal;
      flagSignal = false;
    }

    else if (millis() > tmp_separator)
    {
     flagSpace = false;
     flagLetter = flagSignal;
     flagSignal = false;
    }
  }
```

```
  //If letter recieved
  if (flagLetter)
  {
      static uint8_t lettersPrinted = 0;
      for (uint8_t i = 0; i < 43; i++)
        if (currLetter == varMorze[i] && lenLetter == lenMorze[i])
        {
          Serial.print(chrMorze[i]);
          lcd.print(chrMorze[i]);
          lettersPrinted++;
        }
      if(flagSpace)
      {
        Serial.print(" ");
        lcd.print(" ");
        lettersPrinted++;
      }
      if (lettersPrinted > 15)
      {
       lcd.scrollDisplayLeft();
      }
    flagLetter = (lenLetter = (currLetter = (flagSpace = 0)));
  }
}
```


A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
1
2
3
4
5
6
7
8
9
0