Dr. Bob Davidov

Подключение периферии к среде разработки систем управления МатЛАБ

Цель работы: Рассмотрение вопросов подключения периферийных устройств к МатЛАБ через интерфейсы компьютера Raspberry Pi и контроллера Arduino UNO.

Задача работы: Построение канала связи МатЛАБ с внешними устройствами в реальном времени.

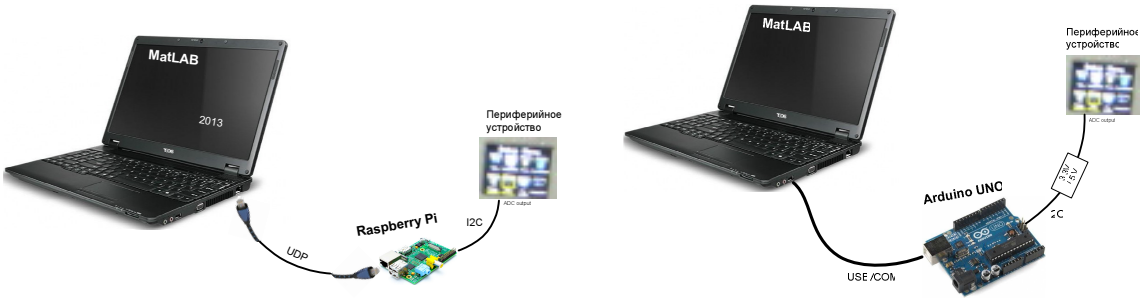
Приборы и принадлежности: Компьютер Raspberry Pi (версия B), контроллер Arduino UNO, МатЛАБ, Simulink, Маршрутизатор ASUS RT-N10P, преобразователи уровней 3.3/0В в -12/+12В и 3.3/0В в 5/0В.

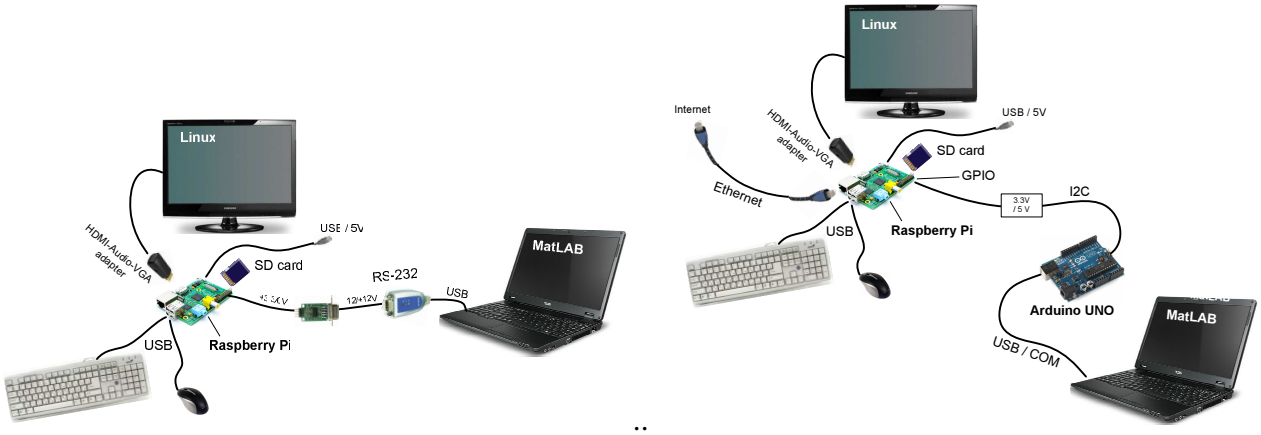
ВВЕДЕНИЕ

Многие устройства имеют цифровые порты ввода вывода и различные последовательные каналы связи, например, распространенный интерфейс I2C. С другой стороны МатЛАБ имеет средства для обмена данными через сетевые устройства или по каналу RS-232. Подключение периферийных устройств к МатЛАБ с целью идентификации, оптимизации, разработки алгоритма управления, отладки алгоритмов петлевых вычислений и т.д. позволяет обеспечить максимальную адекватность моделей управления и проектировать рабочие системы за минимальное время.

В рамках этой работы рассматриваются следующие вопросы (см. Рис. 1).

- Взаимодействие МатЛАБ с платформой Arduino UNO
- Взаимодействие МатЛАБ с компьютером Raspberry Pi
- Взаимодействие Raspberry Pi с контроллером Arduino UNO
- Взаимодействие МатЛАБ с периферийными устройствами через I2C интерфейсы Raspberry Pi и Arduino UNO.
- Работа с Raspberry Pi в режиме консоли с удаленного компьютера.





..

Рис. 1. Варианты подключения периферии к МатЛАБ.

ОБЩИЕ СВЕДЕНИЯ

Взаимодействие Arduino UNO с периферийными устройствами через I2C интерфейс

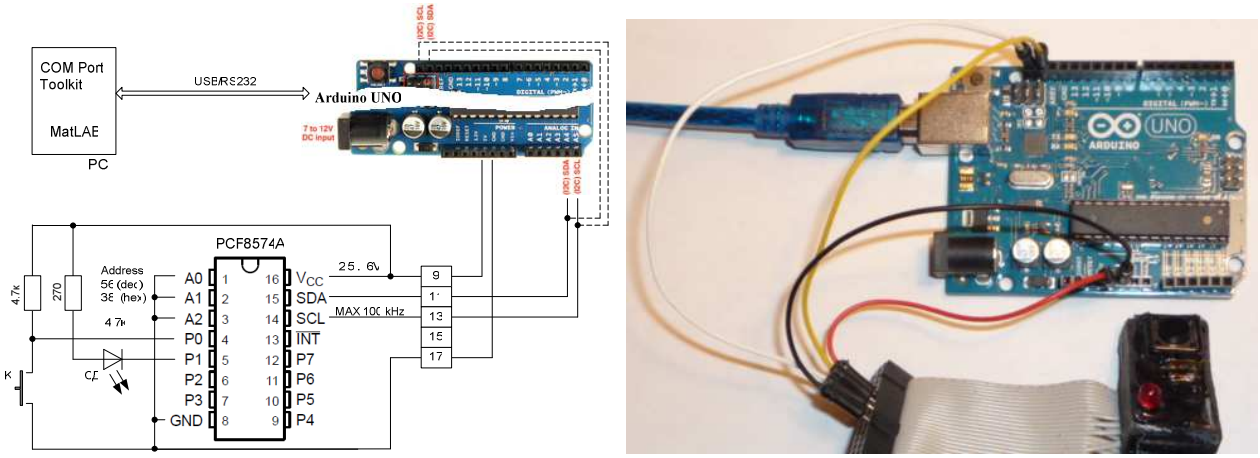


Рис. 2. Вариант подключения цифрового порта PCF8574A к Arduino через I2C последовательный канал. В режиме I2C обозначенные пары контактов SDA и SCL контроллера Arduino работают параллельно. Устройства можно подключать к любой паре не меняя программу.

Программа чтения состояния кнопки и передача кода состояния в COM порт каждые 0.5 секунды:

```
#include <Wire.h>

#define port_address 0x38 // (0x38 = 56)

void setup()
{
  Wire.begin();                    // join i2c bus in Master mode
```

```
  // Wire.begin(port_address);          // join i2c bus in Slave mode

  Serial.begin(9600);  // start COM with 9600 baud rate
}

void loop()
{
  Wire.requestFrom(56, 1);    // request byte from device #56 (0x38)
   char c = Wire.read();          // receive a byte as character
   Serial.print(c);               // send byte into COM port
   delay(500);                        // delay half second
}
```

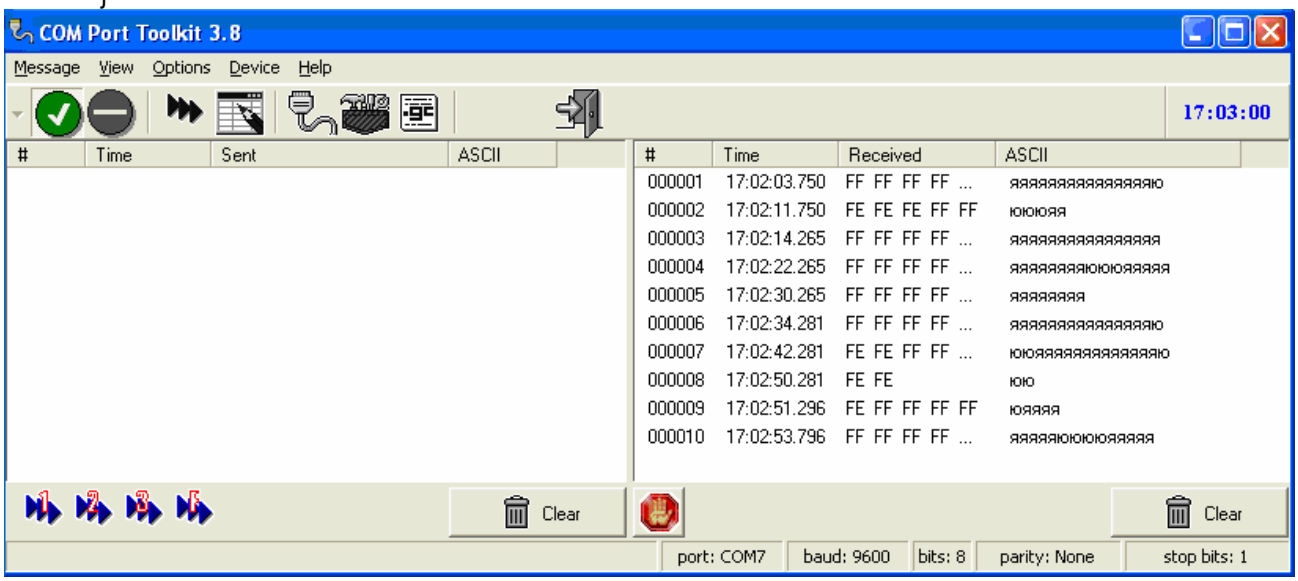


Рис. 3. Отображение символов поступаемых от Arduino через COM порт. Коды FF/FE (символы я/ю таблицы ASCII) отслеживают разомкнутое/замкнутое состояния кнопки.

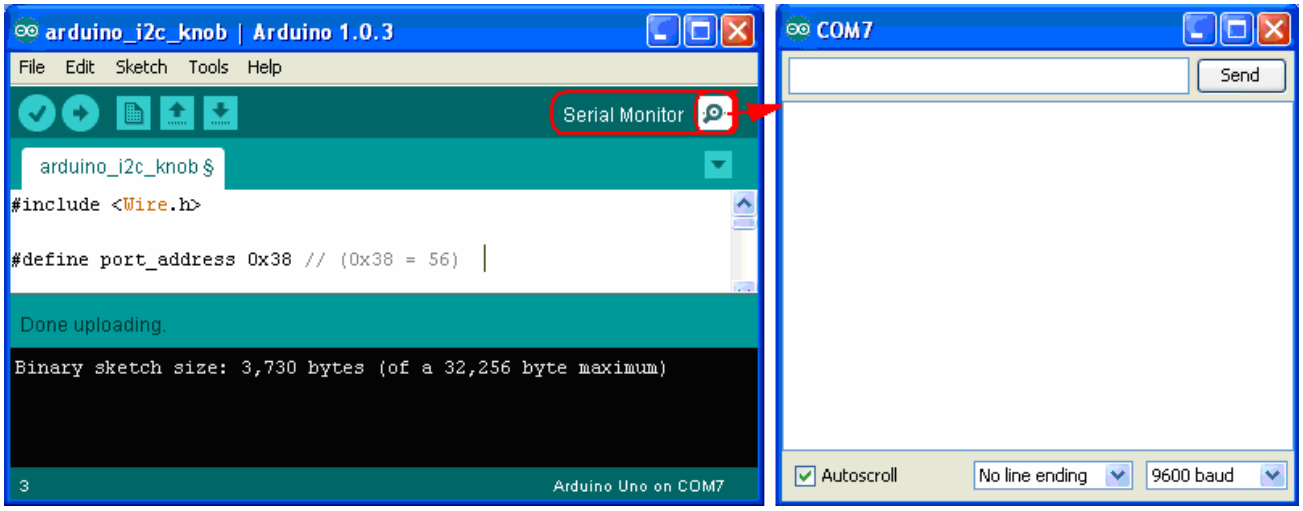


Рис. 4. Отображение данных COM порта можно наблюдать и в окне “Serial Monitor” Arduino который открывается нажатием мыши на выделенном значке.

Программа переключения светодиода каждые 0.5 секунды (file: arduino_i2c_LED):

```
#include "Wire.h"
#define port_address 0x38 // (0x38 = 56)

int led = 13; // give name to Arduino Pin 13 with LED:

void setup() {
  pinMode(13, OUTPUT);
  Wire.begin(port_address);     // initialize i2c
}

void loop() {
  Wire.beginTransmission(port_address);
  Wire.write(0);
  Wire.endTransmission();
  digitalWrite(led, LOW);
  delay(500);

  Wire.beginTransmission(port_address);
  Wire.write(2);
  Wire.endTransmission();
  digitalWrite(led, HIGH);
  delay(500);
}
```

Взаимодействие МатЛАБ с периферийными устройствами через I2C интерфейс контроллера Arduino UNO

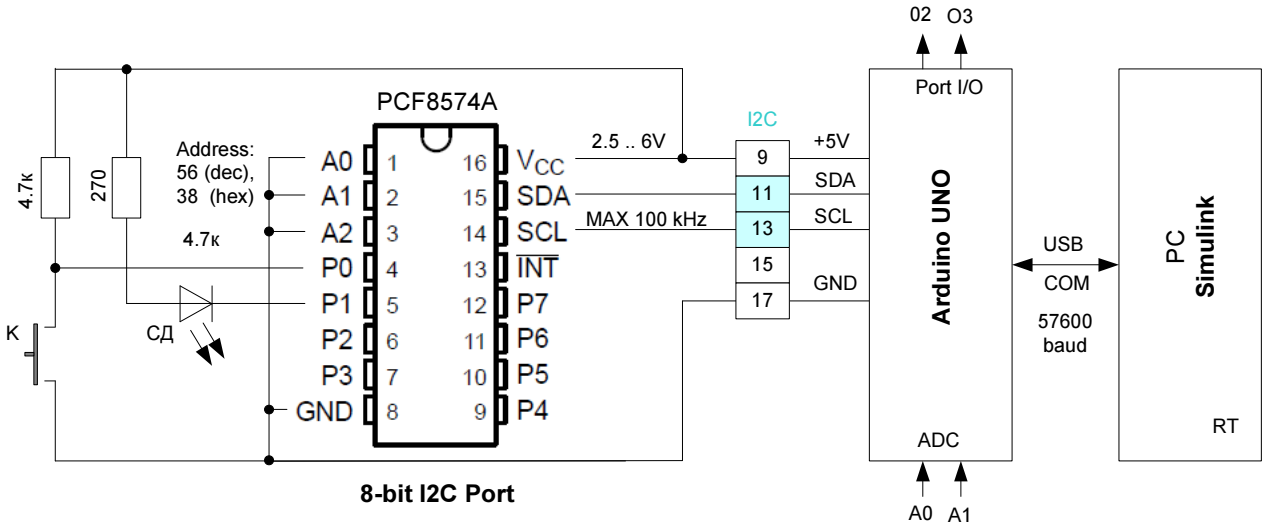


Рис. 5. Структура взаимодействия модели Simulink с цифровым портом и АЦП Arduino UNO, кнопкой и светодиодом, подключенными к цифровому порту с I2C интерфейсом. Модель наблюдает за состоянием периферии и управляет выходами в реальном времени с частотой 10 Гц. Структура не требует специальной компиляции (Ctrl+B) для работы в реальном времени.

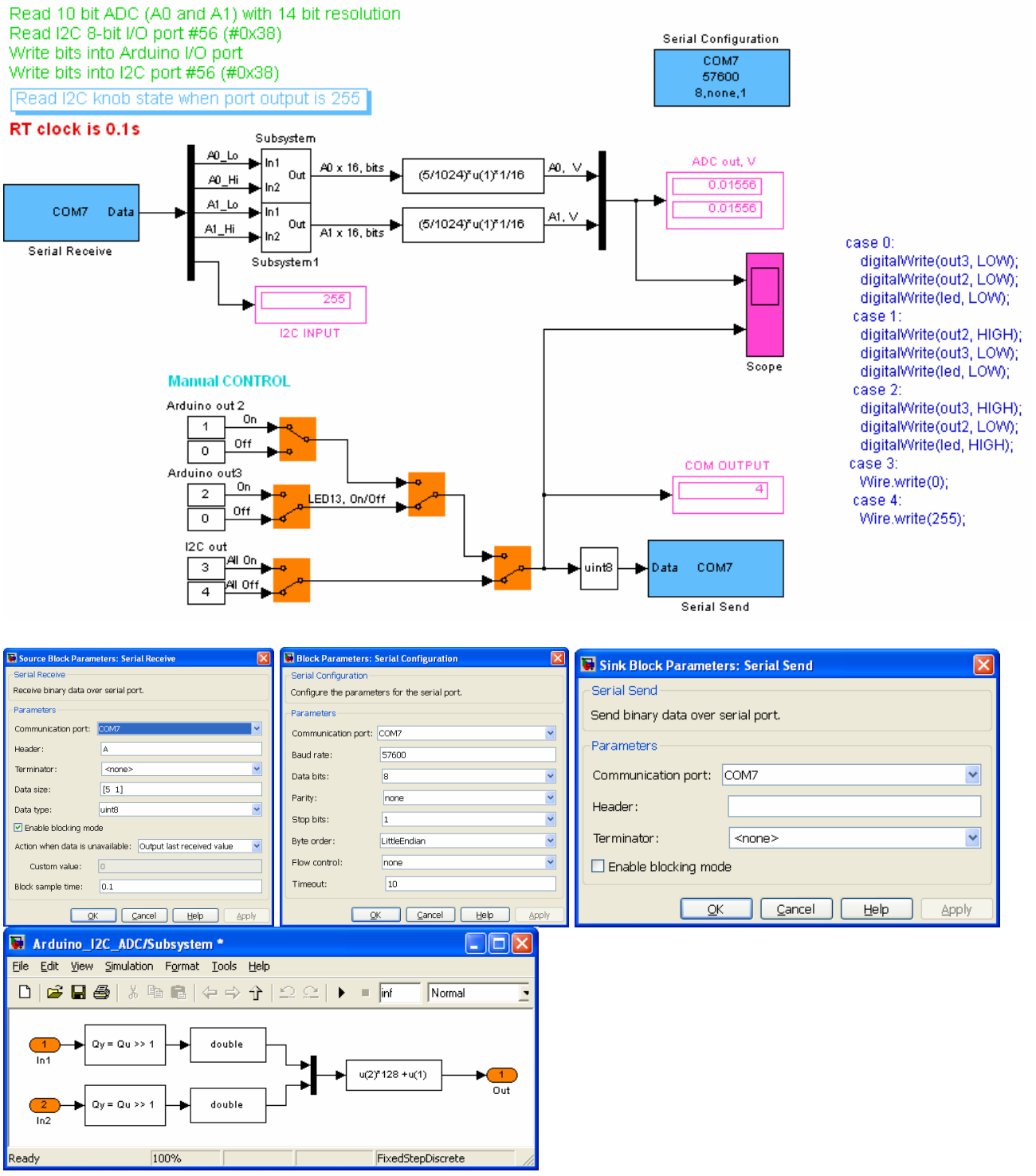


Рис. 6. Модель Simulink и ее блоки.

Программа контроллера Arduino UNO для чтения и обработки кода (см. в программе инструкции case) модели Simulink в момент поступления кода, накопления и усреднения данных АЦП, передачи Simulink состояния I2C кнопки и усредненных значений АЦП каждые 100 мсек:

```
/*
 Operates under Simulink Arduino_I2C_ADC.mdl control
 Reads ADC A0 and A1
 Reads I2C 8-bit I/O port
 Writes bits into Arduino I/O port
 Writes bits into I2C 8bit I/O port
 RT clock is 0.1s

 created 10 January 2013
 by Bob Davidov

 This example code is in the public domain.
 */

#include "Wire.h"

#define port_address 0x38 // (0x38 = 56)

byte ctrl_num;  // control num of 0 .. 255 from simulink
unsigned long set_time = 0;

int i2c_in;

int out3 = 3; // control Pin #3
int out2  = 2; // control Pin #2
int led = 13; // control LED

int adc_A0;
int adc_A1;

unsigned long adc_A0_sum = 0;
unsigned long adc_A1_sum = 0;
int loop_num = 0;

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(57600); //300, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200
  Serial.flush();  // clear input buffer

//  Wire.begin(port_address);     // initialize i2c in Slave mode
  Wire.begin();     // initialize i2c in Master mode

  pinMode(out3, OUTPUT);
  pinMode(out2, OUTPUT);
  pinMode(led, OUTPUT);

//  digitalWrite(led, LOW);   // turn the LED on (HIGH is the voltage level)
```

```
}

void loop() {
  // get any incoming bytes:
  if (Serial.available() > 0) {
    ctrl_num = Serial.read(); // 0 .. 255 from simulink
    switch (ctrl_num) {
    case 0:
      digitalWrite(out3, LOW);
      digitalWrite(out2, LOW);
      digitalWrite(led, LOW);
      break;
    case 1:
      digitalWrite(out2, HIGH);
      digitalWrite(out3, LOW);
      digitalWrite(led, LOW);
      break;
    case 2:
      digitalWrite(out3, HIGH);
      digitalWrite(out2, LOW);
      digitalWrite(led, HIGH);
      break;
    case 3:
      Wire.beginTransmission(port_address);
      Wire.write(0);
      Wire.endTransmission();
      break;
    case 4:
      Wire.beginTransmission(port_address);
      Wire.write(255);
      Wire.endTransmission();
      break;
    default:
      digitalWrite(led, LOW);
      break;
    }
  }

  adc_A0 = analogRead(0);
  adc_A1 = analogRead(1);

  adc_A0_sum = adc_A0_sum + adc_A0;
  adc_A1_sum = adc_A1_sum + adc_A1;

  loop_num = loop_num + 1; // ~ 0 .. 180 lopps per 0.1 ms

  // send to COM port results
```

```
  unsigned long time = millis();
  if (time > set_time) {
    set_time = set_time + 100;

    adc_A0 = (adc_A0_sum * 16) / loop_num;
    adc_A1 = (adc_A1_sum * 16) / loop_num;

    byte adc_A0_Hi = ((adc_A0 >> 6) & 0xFE);
    byte adc_A0_Lo = ((adc_A0 << 1) & 0xFE);

    byte adc_A1_Hi = ((adc_A1 >> 6) & 0xFE);
    byte adc_A1_Lo = ((adc_A1 << 1) & 0xFE);

    Serial.print("A"); // it is header
    Serial.write(adc_A0_Lo);
    Serial.write(adc_A0_Hi); // output byte: uint8

    Serial.write(adc_A1_Lo);
    Serial.write(adc_A1_Hi);

    Wire.requestFrom(56,2);   // request byte from i2c device #56 (0x38)
    i2c_in = Wire.read();     // receive a byte as character

    Serial.write(i2c_in);     // send i2c input byte

    adc_A0_sum = 0;
    adc_A1_sum = 0;

    loop_num = 0;
  }
}
```

Взаимодействие МатЛАБ с компьютером Raspberry Pi через RS-232 интерфейс

Рис. 7.Разъем DE-9, часто используемый для передачи по протоколу RS-232

Таблица. Сигналы COM порта.

Номер контакта	Тип сигнала	Пояснения
2	RxD Receive Data	Прием данных
3	TxD Transmit Data	Передача данных
5	Signal ground	

Raspberry Pi имеет канал последовательной приемо-передачи данных конфигурируемый на базе порта GPIO. Для подключения к GPIO с 3.3 В логикой устройства с RS-232 интерфейсом и уровнями +/- 12 В необходимо согласовать уровни напряжения двух устройств, например, при помощи микросхемы MAX3232, как это показано на Рис. 8.

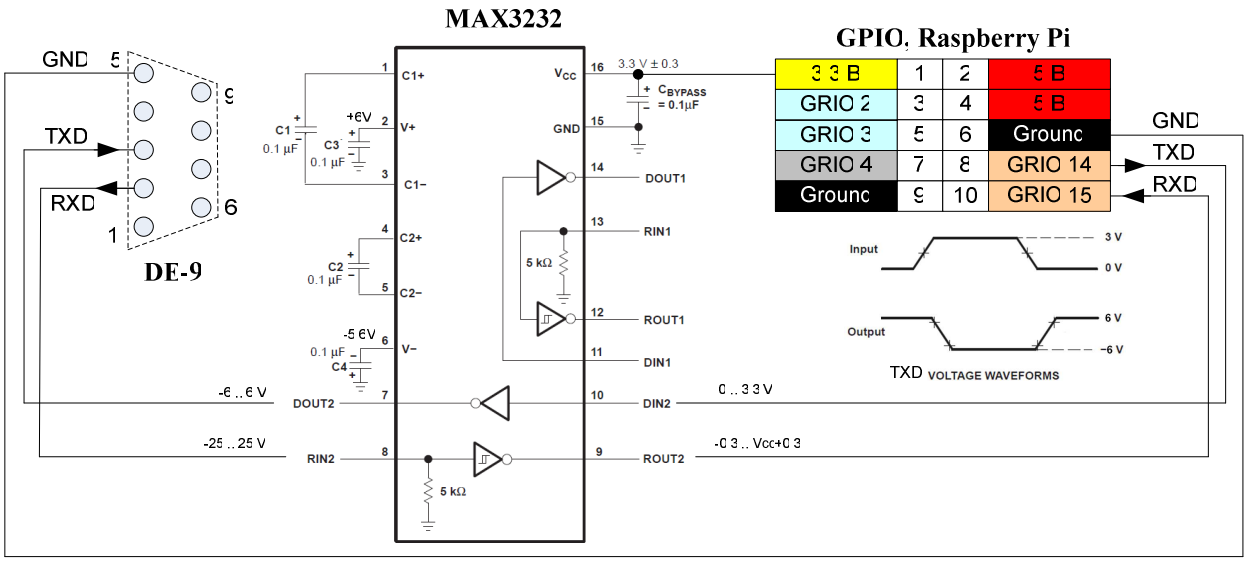


Рис. 8.Схема согласования уровней напряжения порта GPIO (0/3.3 В) и устройства с RS – 232 интерфейсом (-12/+12 В), подключаемого к разъёму DE-9.

Для соединения двух устройств через COM порты по каналу RS – 232 можно использовать нуль-модемный кабель (см. Рис. 9).

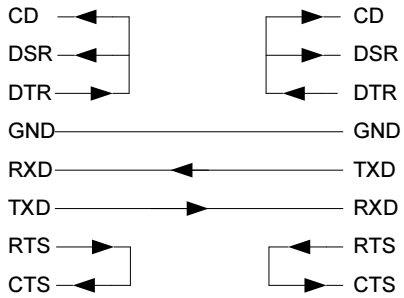


Рис. 9. Схема простейшего нуль-модемного кабеля.

Подключение Raspberry Pi к USB порту компьютера можно выполнить при помощи USB / RS232 преобразователя, например, как показано на Рис. 10.

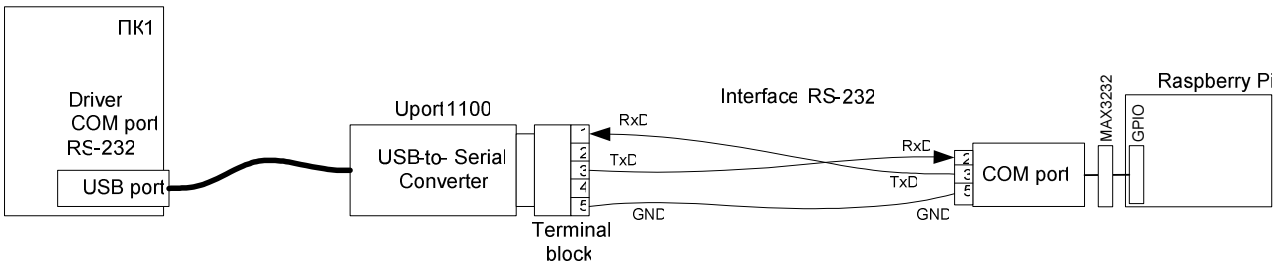


Рис. 10. Схема подключения Raspberry Pi к ПК через USB / COM порты и USB/RS-232 адаптер Uport 1100.

Для работы с последовательным портом необходимо подключить соответствующие библиотеки и выполнить настройки представленные ниже

Установка python serial

sudo apt-get install python-serial

Проверка python serial:

```
$ python
>>> import serial
>>> exit()
```

Для использования последовательного порта необходимо отключить параметр getty поставив # в начало строки T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100 файла /etc/inittab:

#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100

Для приема RXT данных без многочисленного повторного возврата данных в TXD канал следует удалить текст

console=ttyAMA0,115200 kgdboc=ttyAMA0,115200

из файла /boot/cmdline.txt и перезагрузить компьютер

Пример программы проверки передачи данных Raspberry Pi персональному компьютеру.

```
import serial
import time
delay = 0.5
comunicacion = serial.Serial(' /dev/ttyAMA0', 9600)
comunication.open()

while True:
   comunication.write('1')
   time.sleep(delay)
   comunication.write('0')
   time.sleep(delay)
   comunication.write('2')
   time.sleep(delay)
   comunication.write('3')
```

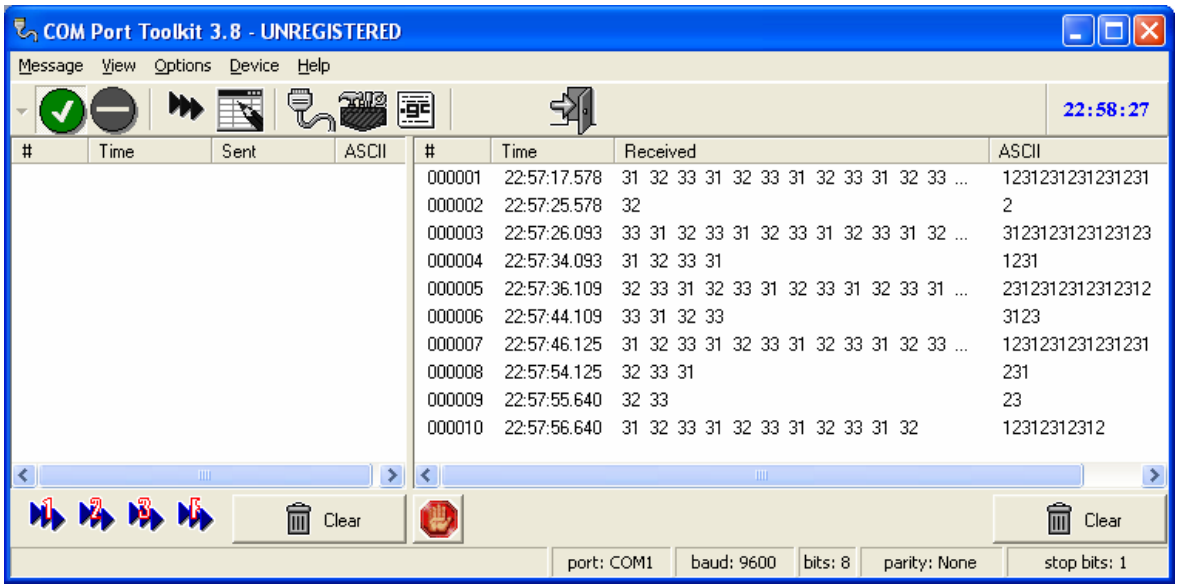


Рис. 11. Проверка приема данных от Raspberry Pi по каналу RS-232.

Пример программы (на языке Python) приема передачи данных по каналу RS-232 (stud_7_8/rs_matlab.py):

```
import time
import serial

# Configure serial port
s = serial.Serial()
s.baudrate = 57600
s.bytesize = 8
s.parity = 'N'
s.stopbits = 1
s.timeout  = 0.5
s.xonxoff = 0
s.rtscts = 0
s.port("/dev/ttyAMA0")

s.open()
s.flush() # clear buffering

def readch(s):
    while True:
        ch = s.read() # read one byte
        if ch=='1' or ch=='2' or ch=='3' or ch=='4':
            return ch

while True:
    rcv = readch(s)
```

```
if rcv == '1':
   s.write('A12345')
if rcv == '2':
   s.write('A23456')
if rcv == '3':
   s.write('A34567')
if rcv == '4':
   s.write('A45678')
if rcv == '5':
   s.write('A')
   i = 25
   s.write(chr(i))  # i = 0 .. 255
   s.write(chr(0x25))
   s.write(chr(255))
   s.write(chr(0xFF))
   s.write(chr(19))

 time.sleep(0.1);
```

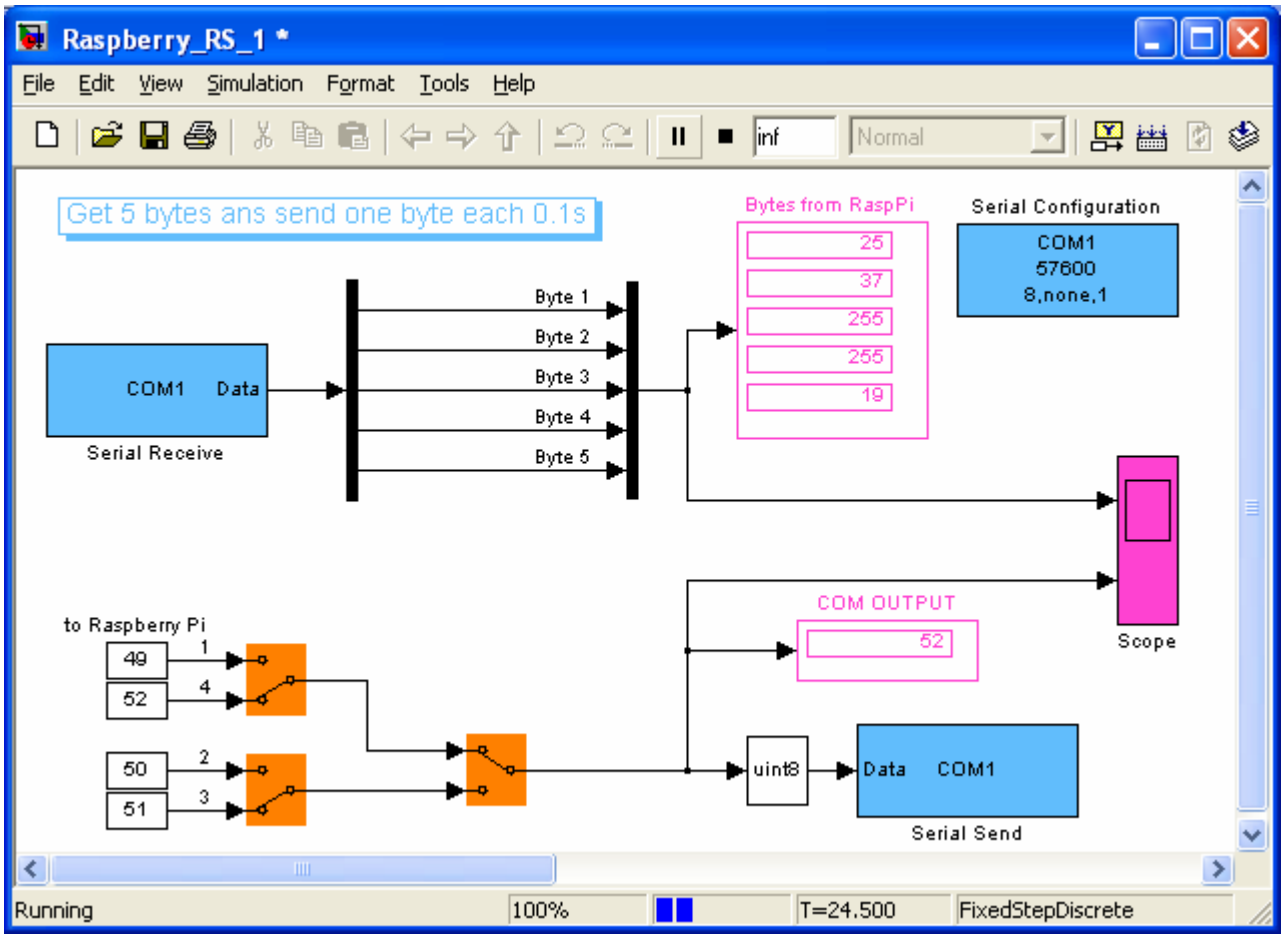


Рис. 12. Модель проверки приема-передачи данных МатЛАБ - Raspberry Pi по каналу RS-232.

Обеспечение временных интервалов в Raspberry Pi

С самого начала Linux не проектировалась как операционная система реального времени. Ее многопользовательские и многозадачные свойства не способствует решению задач реального времени.

Компьютер Raspberry Pi не имеет таймера для отсчета временных интервалов. Системную дату и время можно вычислять внешними аппаратными средствами - “часами”, например, с I2C интерфейсом. Но их односекундное разрешение недостаточно для вычисления временных интервалов с миллисекундной точностью.

Использование собственного аппаратного ШИМ для формирования временных интервалов не годится поскольку ШИМ гарантирует работу с заданной скважностью, но не может обеспечить работу с заданным периодом.

Задержки типа time.sleep(delay) выполняются чисто программными средствами и имеют хорошую точность, но изменение задержки delay на заданную величину в процессе выполнения программы, например, для подстройки частоты - невозможно.

Интерпретатор Python не лучшее средство программирования приложений реального времени (например, в сравнении с C). Можно заметить, что даже небольшая программа построения временных интервалов в цикле делает это с заметными флуктуациями.

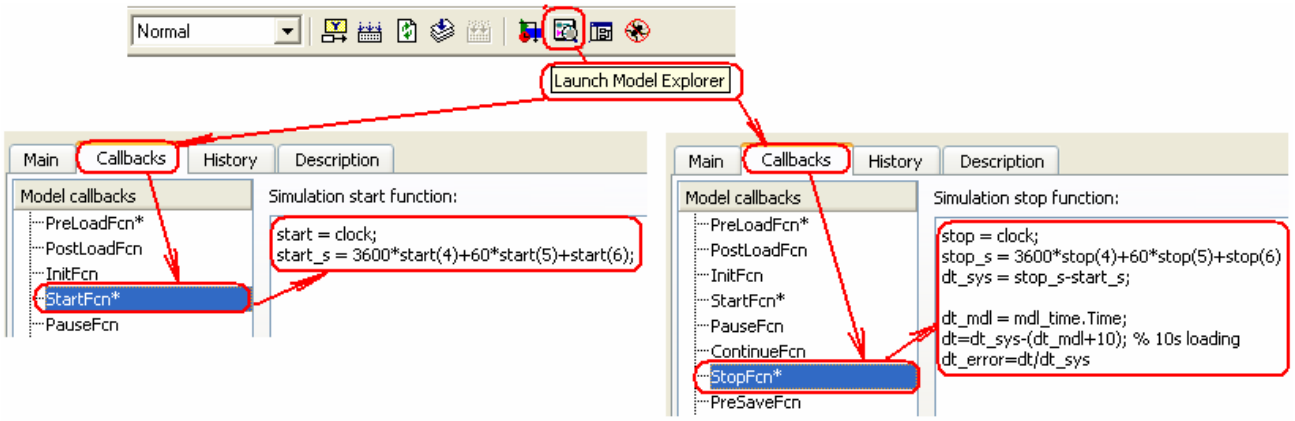
Для обеспечения совместной работы модели Simulink и интерфейса Raspberry в реальном времени можно выполнить следующую подстройку таймера модели по системному таймеру компьютера модели.

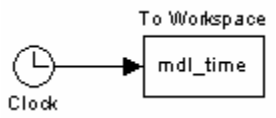
1. Включить в код Raspberry для циклической связи с МатЛАБ задержку, равную заданному периоду вычисления модели, например, 0.1 сек.

   time.sleep(0.1)

2. Дополнить модель simulink следующими средствами для вычисления отклонения виртуального времени выполнения модели от соответствующего системного времени, с учетом задержки запуска модели (~10сек.).

   Примечание. Такт вычисления simulink модели, поддерживаемый средствами RS232 интерфейса, равен времени цикла выполнения программы компьютера Raspberry Pi.





3. Запустить модель на выполнение и течении нескольких минут и после остановки модели найдите в workspace dt_error - отклонение времени модели.
4. Для уменьшения ошибки формирования такта модели установите новую задержку в программе Raspberry: 0.1s (1+dt_error)
5. Проверьте точность временных интервалов вычисления модели (см. п. 3).

Взаимодействие Raspberry Pi с контроллером Arduino по каналу I2C

I²C использует две двунаправленные линии, подтянутые к напряжению питания - последовательная линия данных (SDA, англ. Serial DAta) и последовательная линия тактирования (SCL, англ. Serial CLock).

Классическая адресация включает 7-битное адресное пространство с 16 зарезервированными адресами. Это означает до 112 свободных адресов для подключения периферии на одну шину.

Основной режим работы — 100 кбит/с;

Процедуру обмена начинает ведущее устройство (Master) которое формирует стартовый бит переводом линии SDA из высокого состояния в низкое при высоком уровне линии SCL.

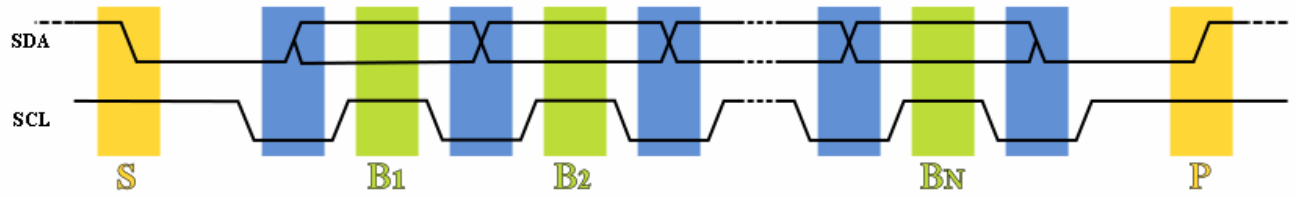


Рис. 13. Тактирование передачи данных по шине I2C [1]. S – стартовый бит (спад SDA при высоком SCL) ; B1, B2, …, B_N – биты данных (бит начинает при низком SCL и удерживается при высоком SCL); P – стоповый бит (подъём SDA при высоком SCL).

После формирования стартового бита, ведущий переводит линию SCL в низкое состояние и выставляет на линию SDA старший бит первого байта сообщения. Количество байт в сообщении не ограничено. Изменение данных на линии SDA выполняется только при низком уровне сигнала на линии SCL. Данные не должны изменяться во время высокого состояния синхроимпульса SCL.

Процедура обмена завершается стоповым битом - переходом линии SDA из низкого состояния в высокое при высоком состоянии линии SCL.

После приема байта данных (восьми бит) ведомое устройство (Slave) выставляет на шину SDA 9-ый бит подтверждения приема байта.

Ведущий не имеет права на управление переходом линии SCL из низкого состояния в высокое но может удерживать линию SCL в низком состоянии до момента готовности к приему следующего бита.Таким образом скорость передачи любого ведущего адаптируется к скорости медленного устройства.

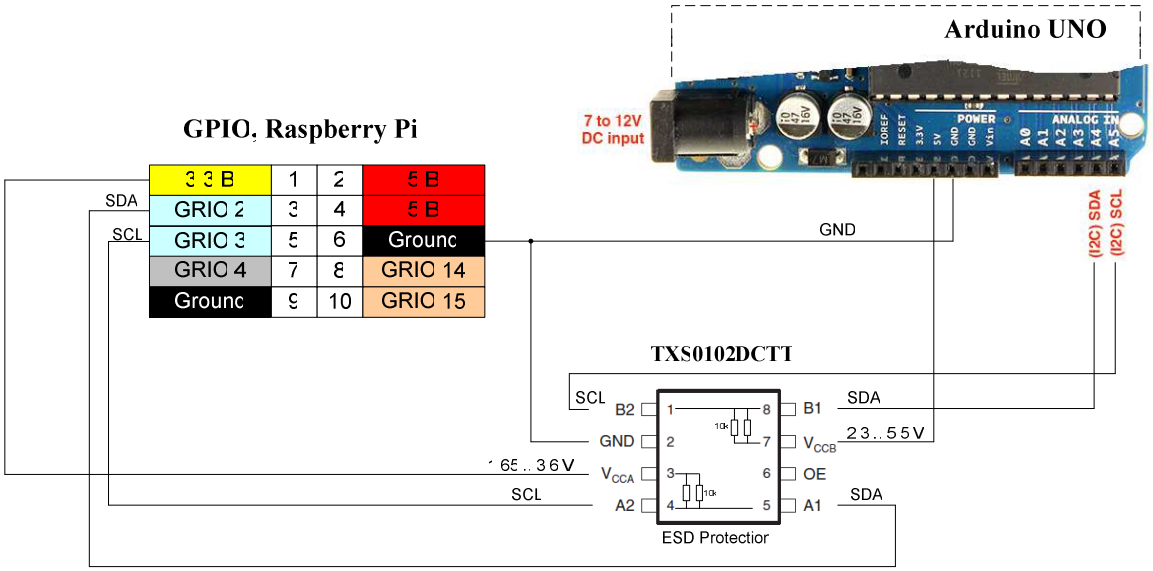


Рис. 14. Пример подключения Arduino - Raspberry Pi по каналу I2C через согласователь напряжения 3.3В/5В TXS0102D.

ПРИМЕРЫ ПОЛУЧЕНИЯ ПРОВЕРЕННЫХ РЕЗУЛЬТАТОВ И ВАРИАНТЫ ДЛЯ САМОКОНТРОЛЯ

Задание 1. Взаимодействие Arduino UNO с периферийными устройствами через I2C интерфейс.

Проверьте работоспособность связи контроллера Arduino UNO с кнопкой и светодиодом, подключенных к цифровому порту PCF8574A с I2C интерфейсом. Схема соединения (Рис.2) и программы Arduino приведены выше в разделе “Взаимодействие Arduino UNO с периферийными устройствами через I2C интерфейс”.

Задание 2. Взаимодействие МатЛАБ с периферийными устройствами через I2C интерфейс контроллера Arduino UNO

.

Проверьте работоспособность связи среды Simulink с периферийными устройствами контроллера Arduino UNO, подключенных к I2C интерфейсу. Схема соединения (Рис.5),

модель Simulink и программа контроллера Arduino приведены выше в разделе “Взаимодействие МатЛАБ с периферийными устройствами через I2C интерфейс контроллера Arduino UNO”.

Задание 3. Построение канала МатЛАБ – Raspberry Pi для работы в реальном времени.

1. Постройте в Simulink модуль, разделяющий амплитуду сигнала на два байта.

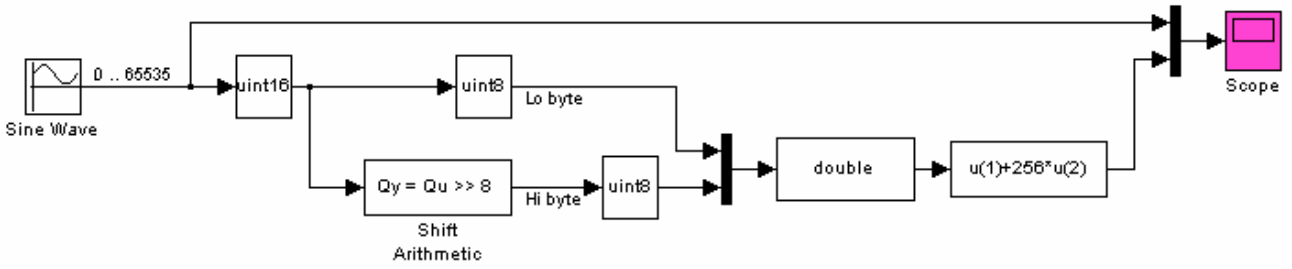


Рис. 15. Модель Simulink.

2. Проверьте работоспособность модуля

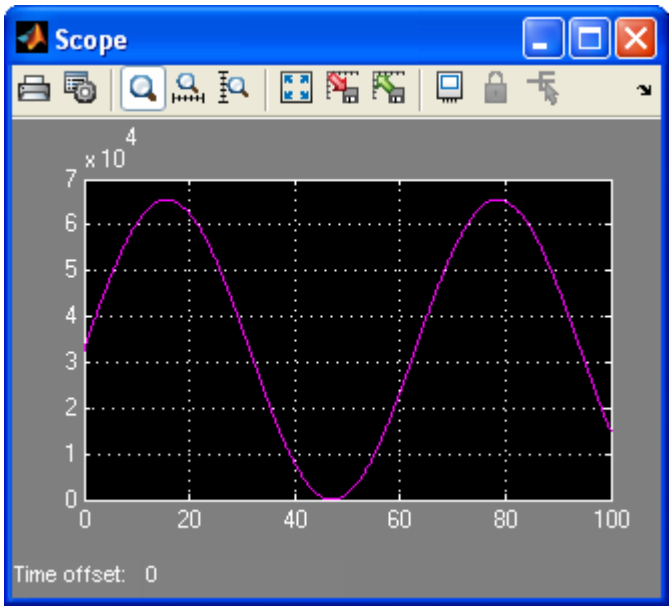


Рис. 16. Окно блока Scope модели Simulink.

3. Создайте программу на языке python для приёма трех байт из канала последовательной передачи данных, восстановления сигнала и передачи сигнала с дополнительными байтами обратно в канал. Принимаемая и передаваема последовательность байт начинается с заголовка ‘A’. Принимаемая и передаваемая амплитуда сигнала состоит из двух байт.

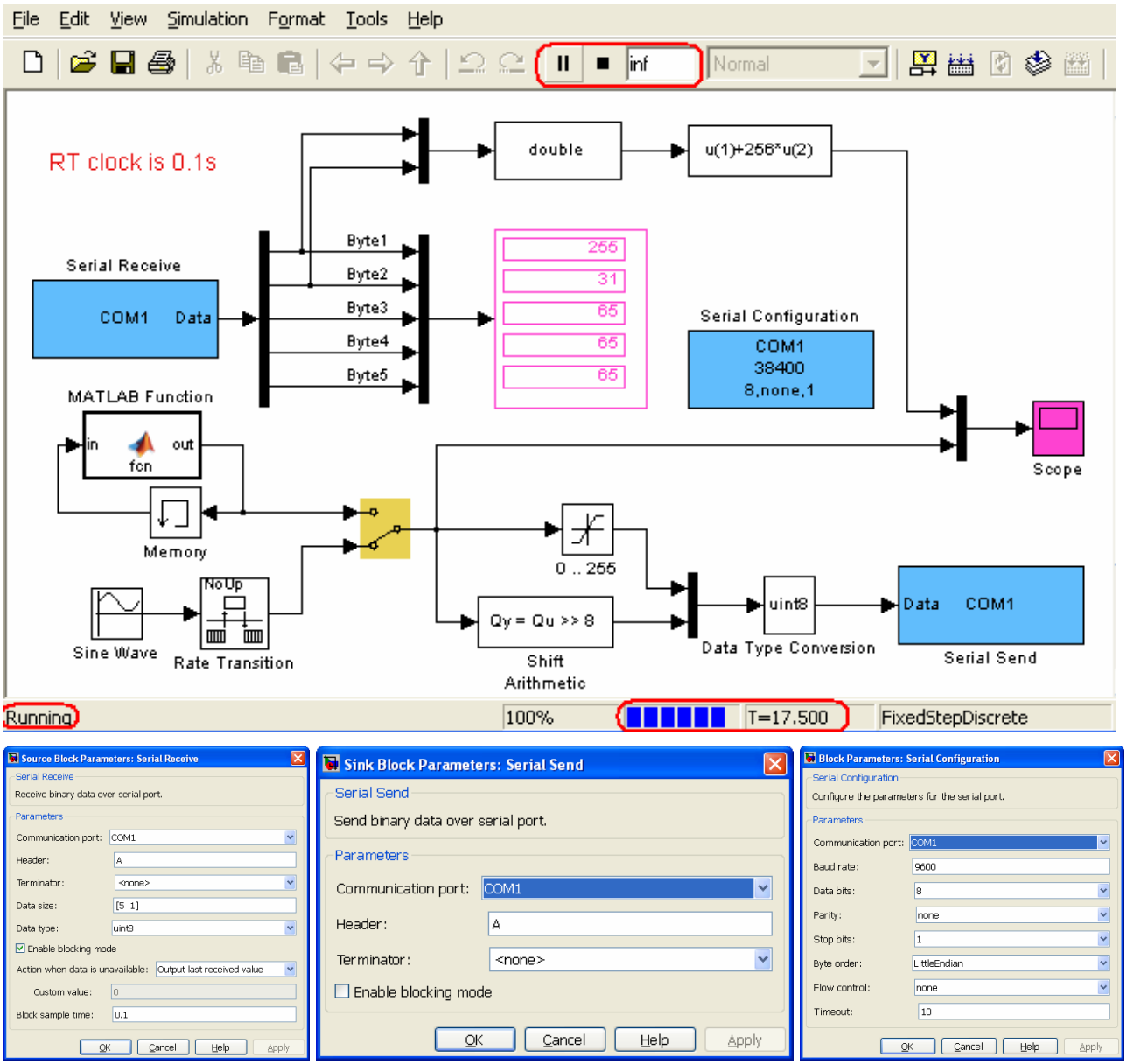
Пример программы:

```
import serial
import time

# Configure serial port
```

```
s = serial.Serial()
s.baudrate = 38400 #9600, 14400, 19200, 38400, 57600, 115200
s.bytesize = 8
s.parity = 'N'
s.stopbits = 1
s.timeout  = 0.5
s.xonxoff = 0
s.rtscts = 0
s.port("/dev/ttyAMA0")

s.open()
s.flush() # clear buffering

def readch(s):
  while True:
    ch = s.read() # read one byte
    if ch=='A':
      rcvlo = s.read()
      rcvhi = s.read()
      rcv = ord(rcvhi)*256 + ord(rcvlo)
      return rcv

while True:
  val = readch(s)
  s.write('A')
  lo = (val & 0000ff)
  hi = (val & 00ff00) >> 8
  s.write(chr(lo))
  s.write(chr(hi))
  s.write('A')
  s.write(chr(0x41))
  s.write(chr(65))
  time.sleep(0.1);

s.close()
```

4. Наберите Simulink модель для связи с Raspberry Pi

Примечание. Поскольку такты времени формируются внешним устройством (Raspberry), то для работы Simulink модели с функциями временной области (например, интеграл, производная, и др.) необходимо сделать следующую установку параметров моделирования: Меню > Simulation > Configuration Parameters > Solver > Tasking and sample time options > Tasking mode for periodic sample time > **Singletasking** или Multitasking




```
function out = fcn(in)
out = in + 1000;
if out < 0 || out > 65000
    out = 0;
end
```

Рис. 17. МатЛАБ модель для проверки связи с Raspberry Pi через RS-232 интерфейс. Частоты приёма-передачи данных МатЛАБ и Raspberry должны совпадать. Скорость передачи выбирается из ряда 300, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200 бод.

5. Проверьте качество связи МатЛАБ – Raspberry

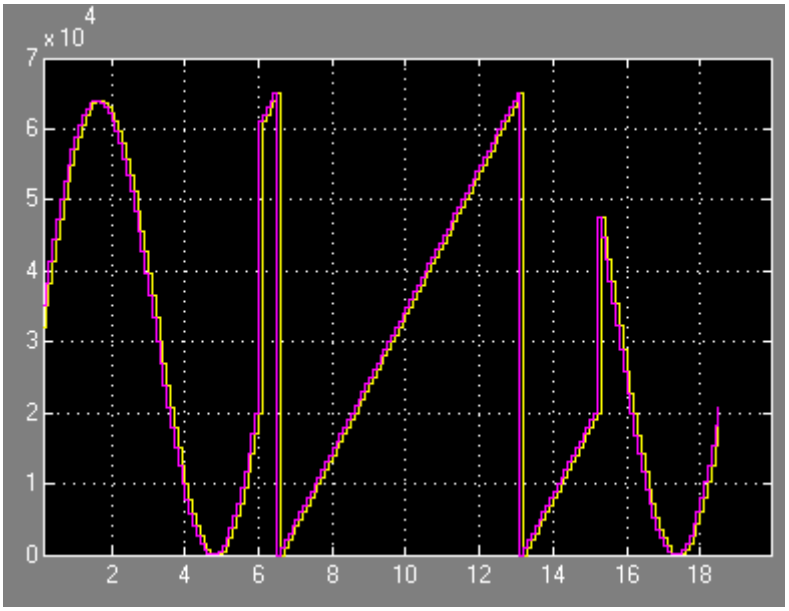


Рис. 18. Зависимость амплитуды пилообразного или синусоидального сигнала от времени. МатЛАБ посылает 2-х байтный пилообразный сигнал (розовый график). Raspberry принимает байты, восстанавливает сигнал, делит восстановленный сигнал на на байты и посылает их МатЛАБ (желтый график) каждую 0.1 сек. МатЛАБ восстанавливает принятый сигнал и сравнивает его с переданным сигналом. Измеренная неточность формирования RT интервала для этой программы составляет 0.1/39 секунды. Задержка выполнения программы МатЛАБ после запуска составляет около 7 секунд.

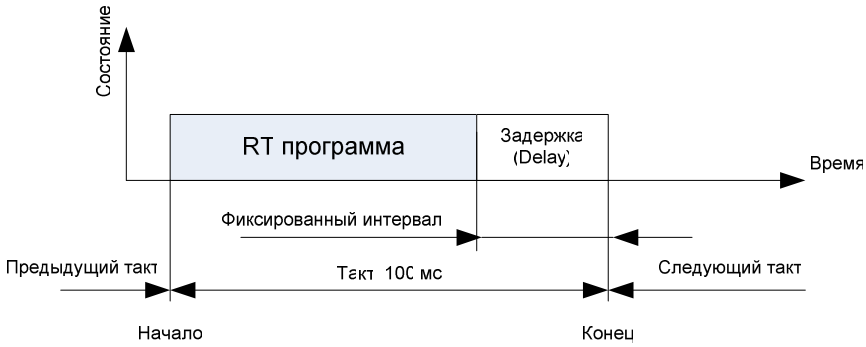


Рис. 19. Формирование такта реального времени с подстройкой задержки (Delay).

6. Создайте программу на языке си для связи Raspberry с МатЛАБ по тому же алгоритму, что и программа на языке Python, представленная в этом задании выше. Точность квантования по времени (100 мсек.) должна обеспечиваться программными средствами без подстройки задержки, как показано на рисунке ниже.

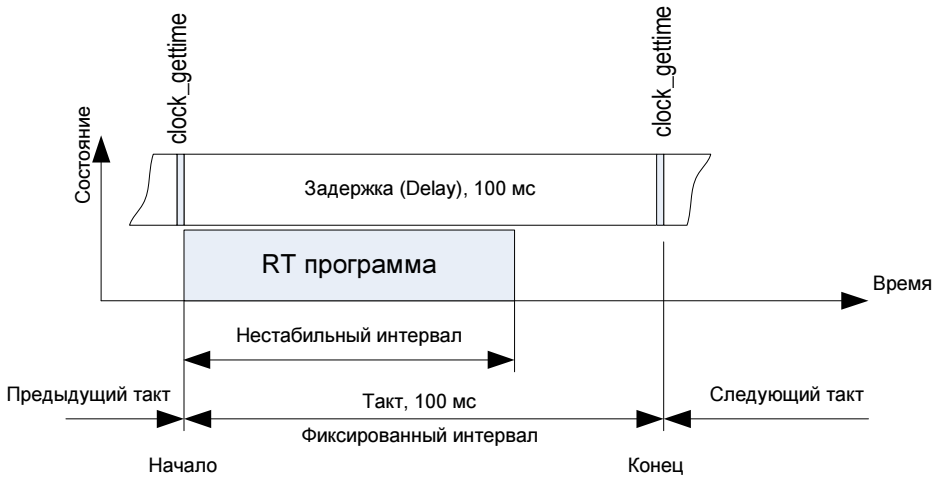


Рис. 20. Формирование фиксированного такта реального времени без с подстройкой задержки.

Пример программы:

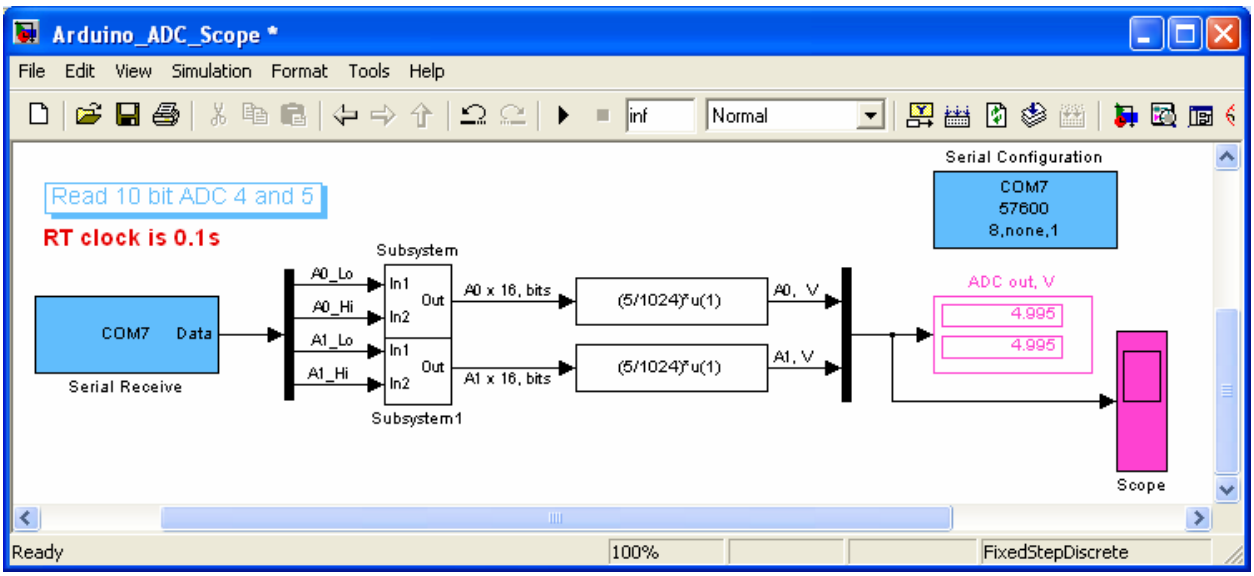
```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <signal.h>
#include <fcntl.h>    // used for UART
#include <termios.h>  // used for UART

/*
#include <string.h>
#include <unistd.h>
#include <time.h>
#include <stdint.h>
#include <semaphore.h>
#include <sys/mman.h>
*/

//static struct timespec timemark;
static int rs232;

static void close_rs232(int unused) // ^C calls close_rs232
{
	close(rs232);
	exit(0);
}

// Adds "delay" nanoseconds to timespecs and sleeps until that time
static void sleep_until(struct timespec *ts, int delay)
{
	ts->tv_nsec += delay;
	if(ts->tv_nsec >= 1000*1000*1000) {
		ts->tv_nsec -= 1000*1000*1000;
		ts->tv_sec++;
	}
	clock_nanosleep(CLOCK_MONOTONIC, TIMER_ABSTIME, ts,  NULL);
```

```
}


// Demo program of RPi and Simulink connection with 100 ms clock
int main(int argc, char **argv)
{
// Setting up the UART
	int rs232 = -1;
	//Open UART
	rs232 = open("/dev/ttyAMA0", O_RDWR | O_NOCTTY | O_NDELAY);
	if (rs232 == -1) {
		printf("Error - Unable open serial port\n");
	}

	//CONFIGURE THE UART
	//The flags (defined in termios.h - see http://pubs.opengroup.org/onlinepubs/007908799/xsh/termios.h.html):
	//  Baud rate:- B1200, B2400, B4800, B9600, B19200, B38400, B57600, B115200, B230400, B460800, B500000,
B576000, B921600, B1000000, B1152000, B1500000, B2000000, B2500000, B3000000, B3500000, B4000000
	tcgetattr(rs232, &cfg);	//get existing configuration setup
	fcntl(rs232, F_SETFL, 0); //fcntl(deviceFD, F_SETFL, FNDELAY);

	////set both incoming and outgoing baud rates...
	cfsetispeed(&cfg, B38400);
	cfsetospeed(&cfg, B38400);
	cfg.c_cflag |= (CLOCAL | CREAD);
	////8N1 (8 data bits, No parity, 1 stop bit)
	cfg.c_cflag &= ~PARENB;
	cfg.c_cflag &= ~CSTOPB;
	cfg.c_cflag &= ~CSIZE;
	cfg.c_cflag |= CS8;
	cfg.c_cflag &= ~CRTSCTS;  //~CNEW_RTSCTS; //disable hardware flow control
	//use RAW unbuffered data mode (eg, not canonical mode)
	cfg.c_lflag &= ~(ICANON | ECHO | ECHOE | ISIG | IGNBRK);
	cfg.c_iflag &= ~(IGNPAR | IXON | IXOFF | IXANY);
	cfg.c_oflag &= ~OPOST;	//raw (unprocessed) output mode

	tcsetattr(rs232, TCSANOW, &cfg);

	struct timespec ts;
	unsigned int delay = 100*1000*1000; // =100 ms (delay in ns)
	unsigned int inbyte1, inbyte2, mlabin;
	unsigned char inbyte, outbytes[6];

	signal(SIGINT, close_rs232);  // ^C calls close_rs232
	while(1)
	{
		while (1)
		{
			if(read(rs232,&inbyte,1)>0)	// read one byte
			{
				if(inbyte=='A')
				{
					read(rs232,&inbyte,1);  // read first byte
					inbyte1 = inbyte;
					read(rs232,&inbyte,1);  // read second byte
					inbyte2 = inbyte;
```

```
                                        mlabin = inbyte1 + inbyte2*255;
                                        break;
                                    }
                                }
                            }

                            sleep_until(&ts,delay);
                            clock_gettime(CLOCK_MONOTONIC, &ts);

                            outbytes[0] = 'A';
                            outbytes[1] = inbyte1;
                            outbytes[2] = inbyte2;
                            outbytes[3] = 10;
                            outbytes[4] = 65;
                            outbytes[5] = 20;
                        write(rs232, outbytes, 6);   // port, bytes to write, number of bytes to write
                    }
                }
```

Задание 4. Построение канала МатЛАБ – Arduino для работы в режиме осциллографа – отображения в Simulink входного напряжения АЦП.

1. Соберите представленную ниже блок схему Simulink для сборки и отображения сигналов принимаемых из COM порта.



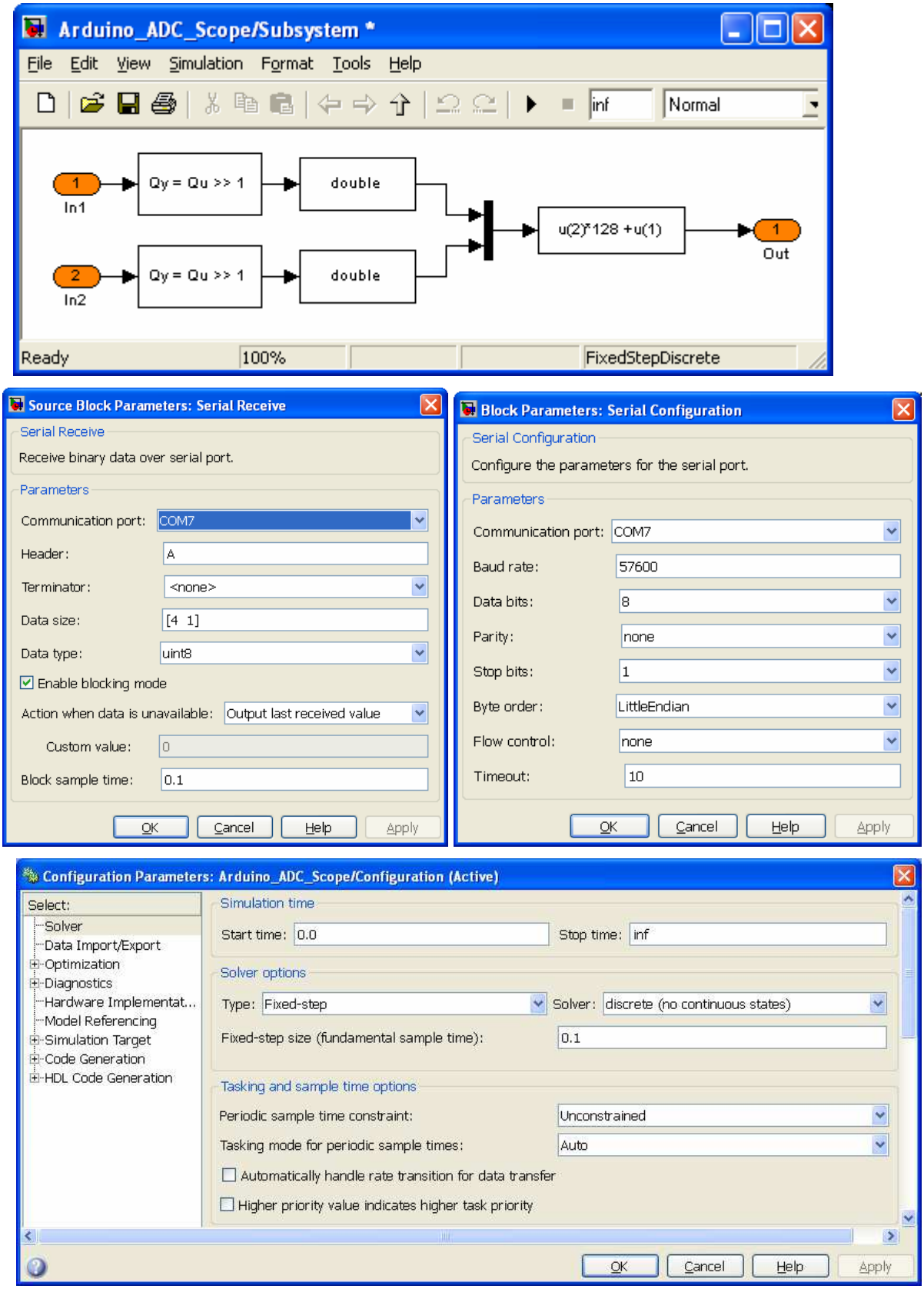


Рис. 21. Параметры модели Simulink.

2. Загрузите в Arduino программу чтения чтения аналоговых сигналов (ADC 4 и ADC 5), разбиения кодов ADC на байты и передачи их с заголовком ‘A’ в COM порт каждые 100 мсек.

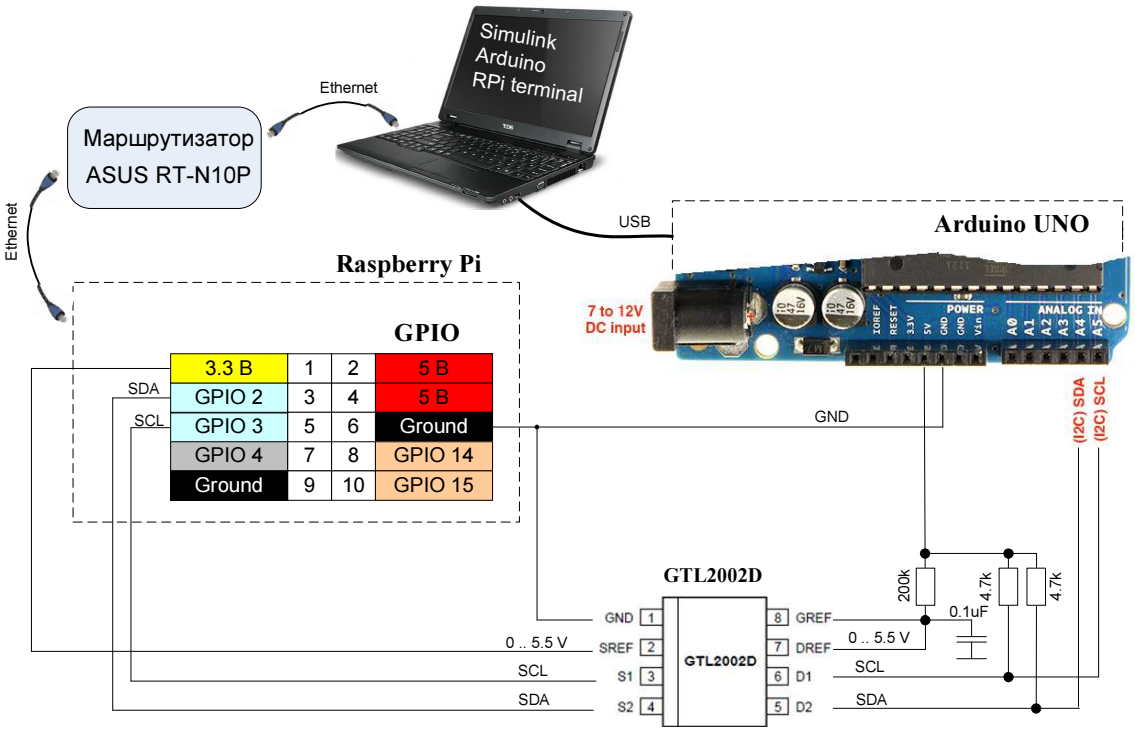
Пример программы:

```
int adc_A4;
int adc_A5;

unsigned long set_time;

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(57600); //300, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200
  Serial.flush();  // clear input buffer

  set_time = millis();
}

void loop() {
  // send to COM port results
  unsigned long time = millis();

  if (time > set_time) {
    adc_A4 = analogRead(4);
    adc_A5 = analogRead(5);

    set_time = set_time + 100;

    byte adc_A4_Hi = ((adc_A4 >> 6) & 0xFE);
    byte adc_A4_Lo = ((adc_A4 << 1) & 0xFE);

    byte adc_A5_Hi = ((adc_A5 >> 6) & 0xFE);
    byte adc_A5_Lo = ((adc_A5 << 1) & 0xFE);

    Serial.print("A"); // it is header
    Serial.write(adc_A4_Lo);
    Serial.write(adc_A4_Hi); // output byte: uint8

    Serial.write(adc_A5_Lo);
    Serial.write(adc_A5_Hi);
  }
}
```

3. Запустите набранную Simulink модель.
4. Подключая 4 и 5 АЦП входы к питанию 3.3 В, 5В и “земле” убедитесь в работоспособности канала цифрового и аналогового отображения АЦП входов в Simulink.

Задание 5. Настройка канала обмена данными Удаленный компьютер – Raspberry Pi – I2C – Arduino UNO – COM - MatLAB.

1. Соберите следующую схему.



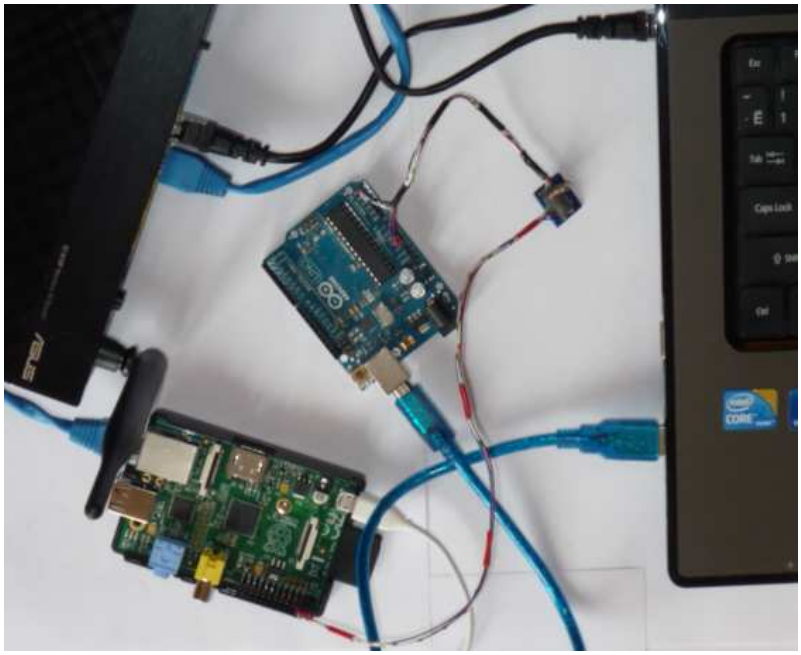


Рис. 22. Схема соединения Терминал – Raspberry Pi – I2C – Arduino – Simulink. Канал I2C RPi конфигурируется по дефолту на контактах GPIO2 (pin 3) и GPIO3 (pin5) после загрузки Raspberry Pi . В Arduino канал I2C конфигурируется на 4 и 5 входах АЦП.

2. Программой IP Scanner определите IP адрес Raspberry Pi

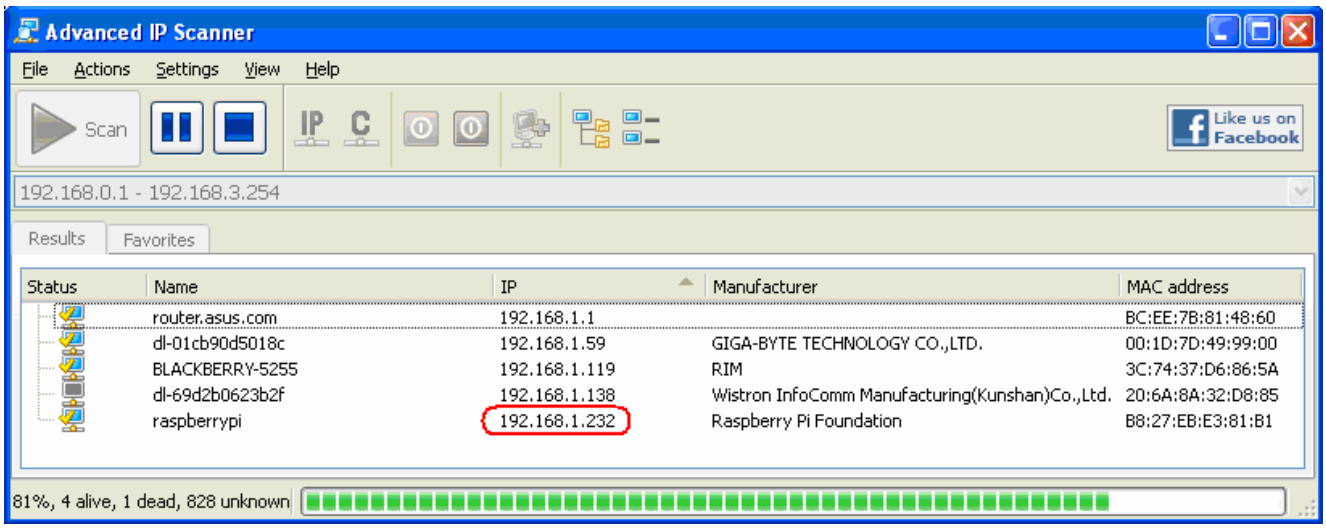


Рис. 23. Сканирование сетевых устройств.

Примечание. Программу Advanced IP Scanner можно использовать без инсталляции.

При необходимости сетевые параметры можно прописать в файле сетевых интерфейсов и конфигурационном файле Raspberry Pi следующим образом.

Файл сетевых интерфейсов Raspberry Pi открывается через консоль командой

sudo nano /etc/network/interfaces.

В файле вместо строки

iface eth0 inet dhcp

следует указать используемые сетевые параметры, например,

iface eth0 inet static
address 10.222.0.182
netmask 255.255.252.0
gateway 10.222.3.254

Конфигурационный файл имеет путь /etc/resolv.conf. Файл resolv.conf обычно содержит IP адреса серверов имён (DNS) которые переводят имена в адрес для любого узла доступного в сети.

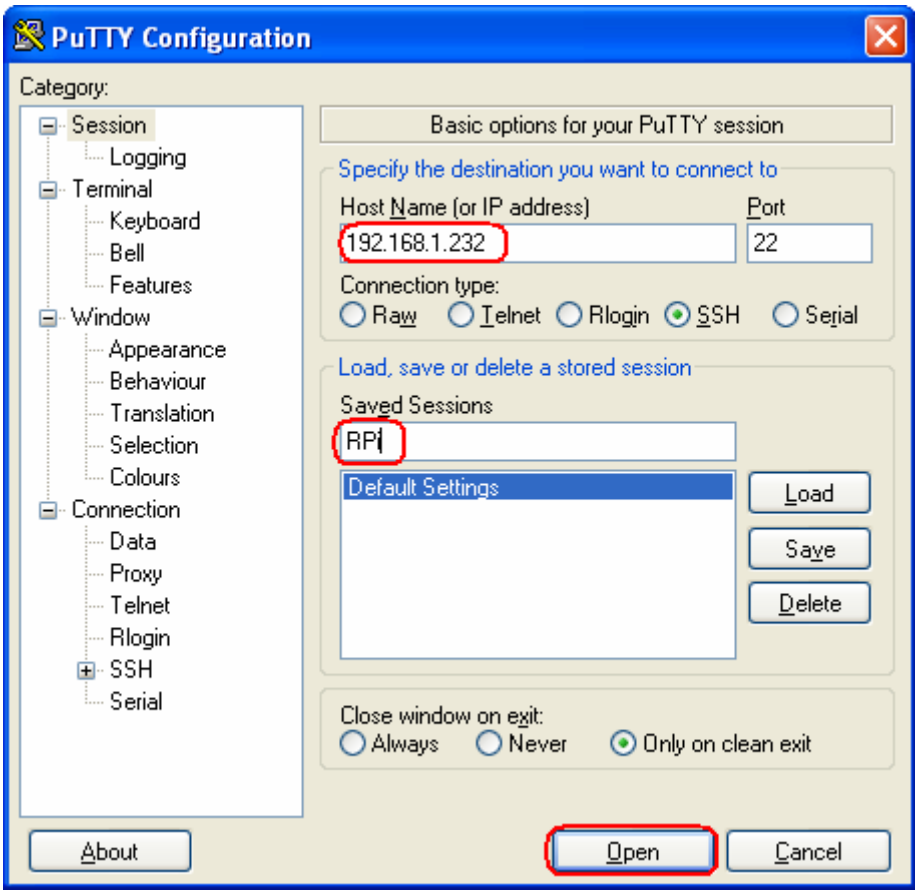
В файле resolv.conf вместо строки

nameserver 8.8.8.8

следует написать адреса DNS сервера, например,

nameserver 83.243.64.2
nameserver 83.243.65.2

3. Программой putty.exe установите связь основного компьютера с Raspberry Pi через терминал.



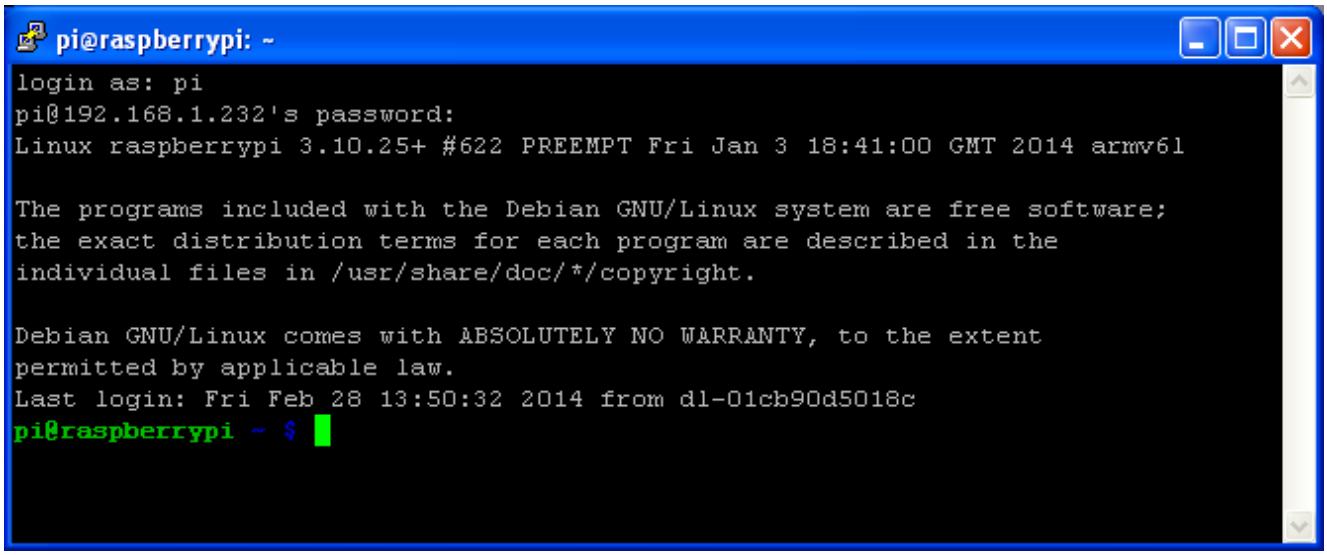


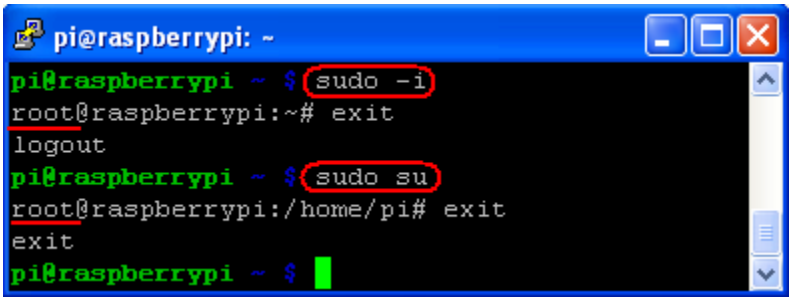
Рис. 24. Запуск удаленного терминала Raspberry Pi с основного компьютера.

Примечание: Для работы в терминале с правами root (например, для того чтобы не набирать слово sudo для выполнения защищенных команд) необходимо ввести команду

sudo –i или

sudo su

Для выхода из режима root необходимо ввести exit.



4. Загрузите средства отображения АЦП входов в среде Simulink, как показано в предыдущем Задании 4.
5. Проверьте соединение Raspberry Pi – Arduino через двунаправленный согласователь уровней 3.3В / 5В. Для этого
6. Загрузите через терминал интерпретатор Python

   $ sudo python
7. В среде интерпретатора импортируйте библиотеку для работы с интерфейсом GPIO

   >>> import RPi.GPIO as GPIO
8. Снимите все конфигурации (включая конфигурацию I2C, устанавливаемую по дефолту)

   >>> GPIO.cleanup()
9. Установите способ нумерации выводов GPIO (BCM или BOARD – номера физических контактов разъема)

   >>> GPIO.setmode(GPIO.BOARD)

10. Сконфигурируйте 3-ий и 5-ый выводы для работы в режиме передачи (OUT) и приёма (IN)

    >>> GPIO.setup(3, GPIO.OUT)

    >>> GPIO.setup(5, GPIO.OUT)

11. Используя следующие команды подавайте на соответствующие выводы GPIO логические единицы и нули. Наблюдайте в Simulink за входами АЦП Arduino которые через преобразователь уровня подключены к 3-му и 5-му контактам GPIO. Убедитесь, что модель Simulink при помощи Arduino отображает состояние выводов GPIO компьютера Raspberry Pi.

    >>> GPIO.output(3, 1)

    >>> GPIO.output(3, 0)

    >>> GPIO.output(5, 1)

    >>> GPIO.output(5,0)

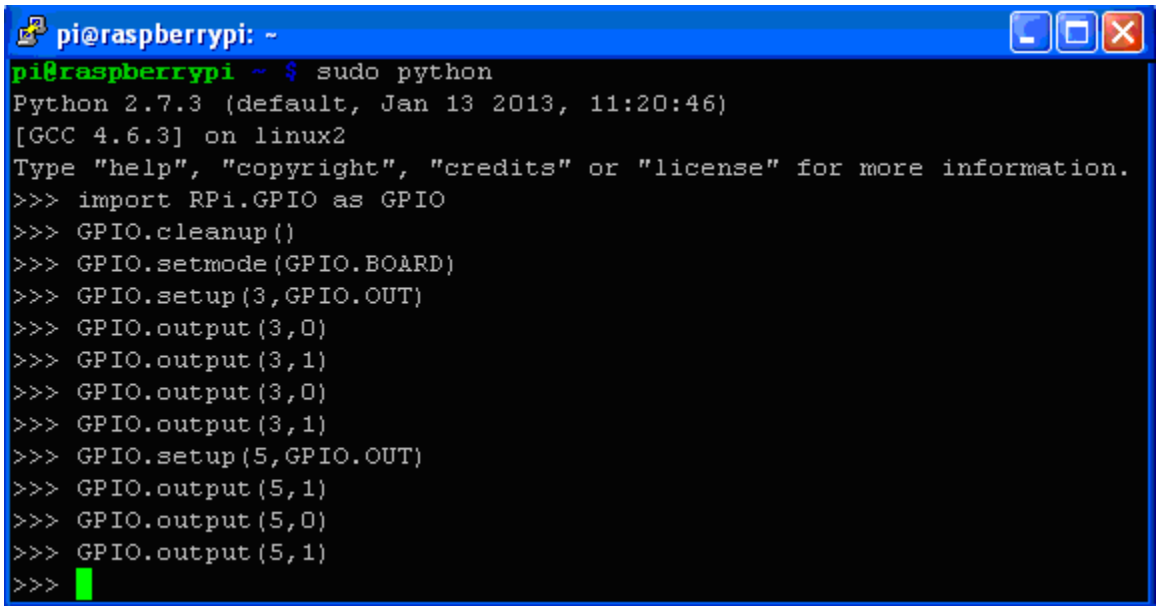


Рис. 25. Настройка и управление выводами GPIO.

12. Верните все выводы GPIO в исходное состояние

    >>> GPIO.cleanup()

13. Завершите проверку соединения Raspberry Pi – Arduino – Simulink и выходом из интерпретатора Python.

    >>> exit()

14. Перезагрузите компьютер RPi

    $ sudo reboot

15. Закройте и снова загрузите putty - терминал Raspberry Pi.

16. Используя следующие команды ОС Linux в разделе /home/pi создайте рабочий каталог, например, st_7_8, и в нем программу передачи данных устройству 0x2A канала I2C.

```
$ dir                    /// вывод содержимого текущей папки
$ pwd                    /// вывод пути текущего каталога
$ sudo mkdir st_7_8      /// создание каталога st_7_8
$ dir                    /// вывод содержимого текущей папки
$ cd st_7_8              /// переход в папку st_7_8
$ sudo nano tsk_4.py     /// вызов редактора nano для создания текстового файла
                         tsk_4.py
```

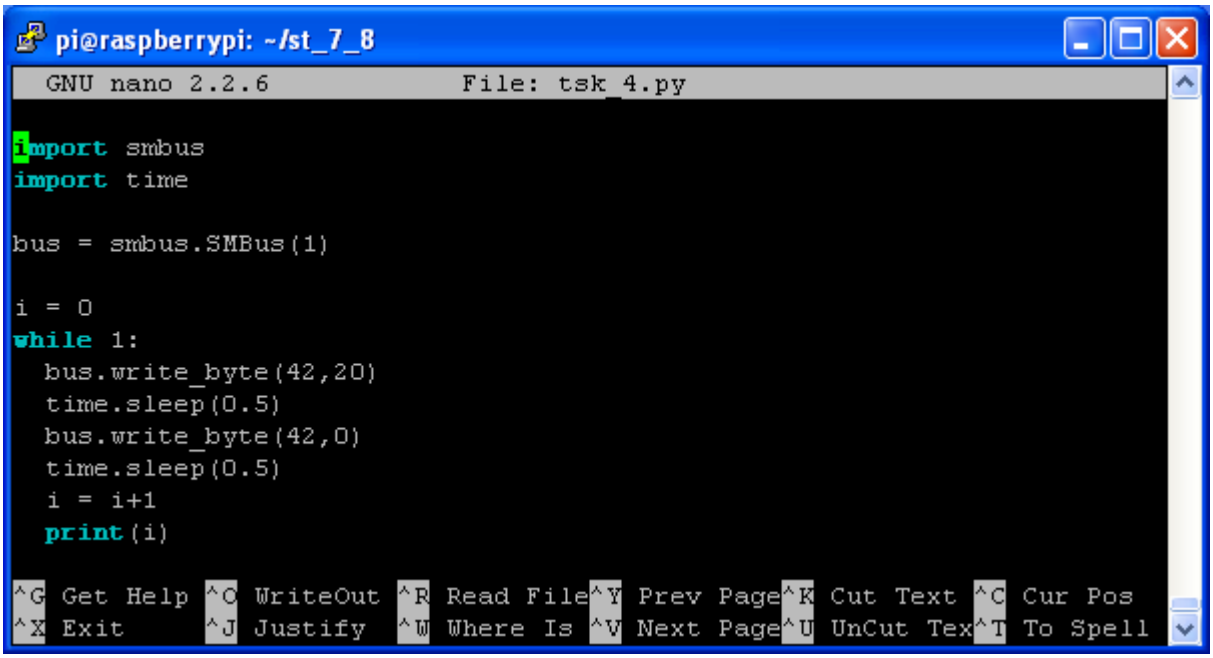


Рис. 26. Окно редактора nano.

```
^X                       /// Выход из редактора

$ sudo modprobe i2c-dev  /// активация I2C интерфейса
$ sudo chmod o+rw /dev/i2c-0

$ i2cdetect –y 1         /// проверка наличия шины “1” канала I2C
```

Рис. 27. Проверка состояния I2C интерфейса: I2C функционирует нормально.

$ dmesg /// проверка частоты I2C канала

$ sudo modprobe -r i2c_bcm2708 && sudo modprobe i2c_bcm2708 baudrate=100000 /// ///установка частоты 100 кГц (при необходимости),

17. Из пакета примеров оболочки Arduino загрузите в контроллер программу приема данных i2c канала (в режиме ведомого – “Slave”) и передачи их в COM порт: \arduino-1.0.3\libraries\Wire\examples\slave_receiver\slave_receiver.ino

Пример программы:

```
#include <Wire.h>
void setup()
{
  Wire.begin(42);              // join i2c bus with address #4
  Wire.onReceive(receiveEvent); // register event
  Serial.begin(9600);          // start serial for output
}

void loop()
{
  delay(100);
}

// function that executes whenever data is received from master
// this function is registered as an event, see setup()
void receiveEvent(int howMany)
{
  while(1 < Wire.available()) // loop through all but the last
  {
    char c = Wire.read(); // receive byte as a character
    Serial.print(c);         // print the character
  }
  int x = Wire.read();    // receive byte as an integer
  Serial.println(x);         // print the integer
}
```

18. Откройте окно для наблюдением за данным COM порта

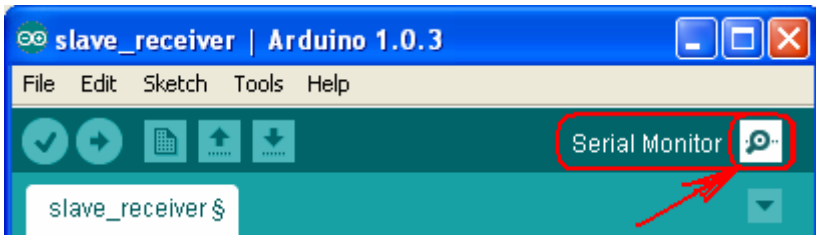


Рис. 28. Открытия окна отображения данных COM канала.

19. Запустите программу Python для передачи данных в i2c канал
    $ sudo python ./tsk_4.py /// запуск программы tsk_4.py под управлением интерпретатора Python

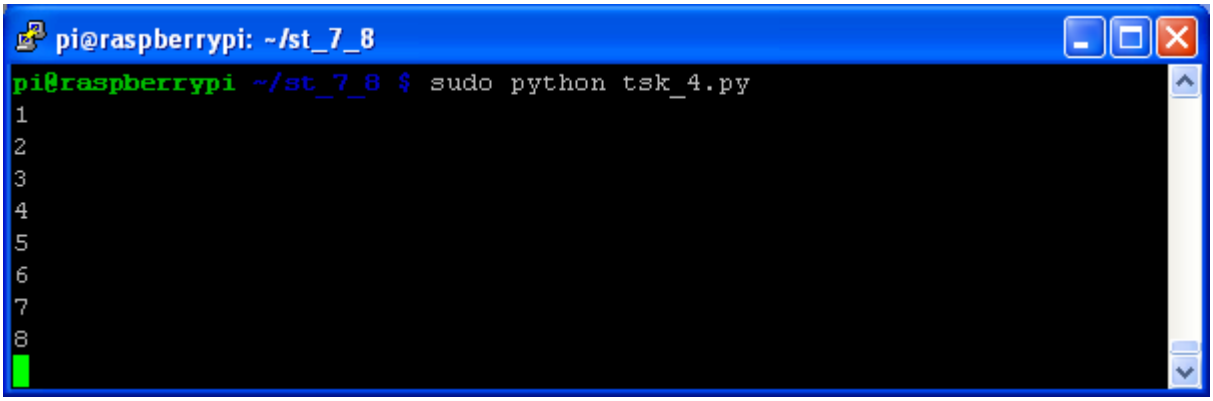


Рис. 29. Запуск и работа Python программы.

20. Наблюдайте в окне COM порта Arduino данные Arduino переданные через I2C интерфейс.

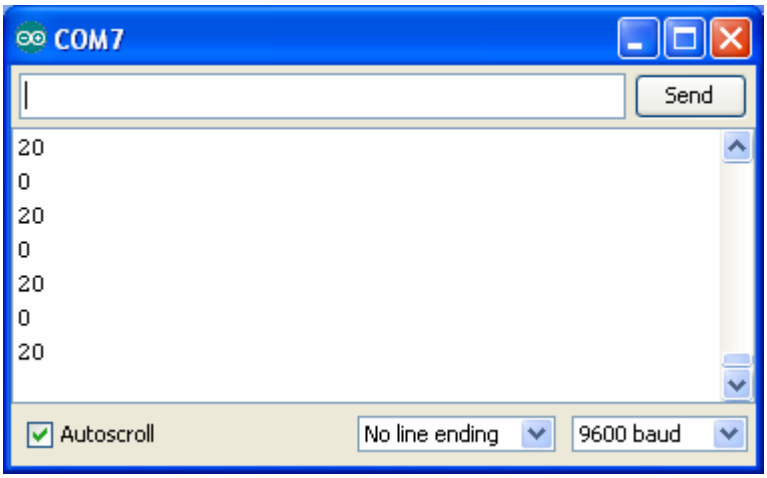


Рис. 30. Данные COM порта.

21. Запустите программу чтения – передачи I2C данных (порт 42) в COM порт со скоростью 57600 бод.

    Пример программы:

```
#include <Wire.h>

void setup()
{
  Wire.begin(42);              // join i2c bus with address #4
  Wire.onReceive(receiveEvent); // register event
  Serial.begin(57600);         // start serial for output
}
```

```
void loop()
{
  delay(100);
}
// function that executes whenever data is received from master
// this function is registered as an event, see setup()
void receiveEvent(int howMany)
{
    char c = Wire.read(); // receive byte as a character
    Serial.print("A"); // it is header
    Serial.print(c);         // print the character
}
```

22. Запустите модель Simulink для приема и отображения данных COM порта

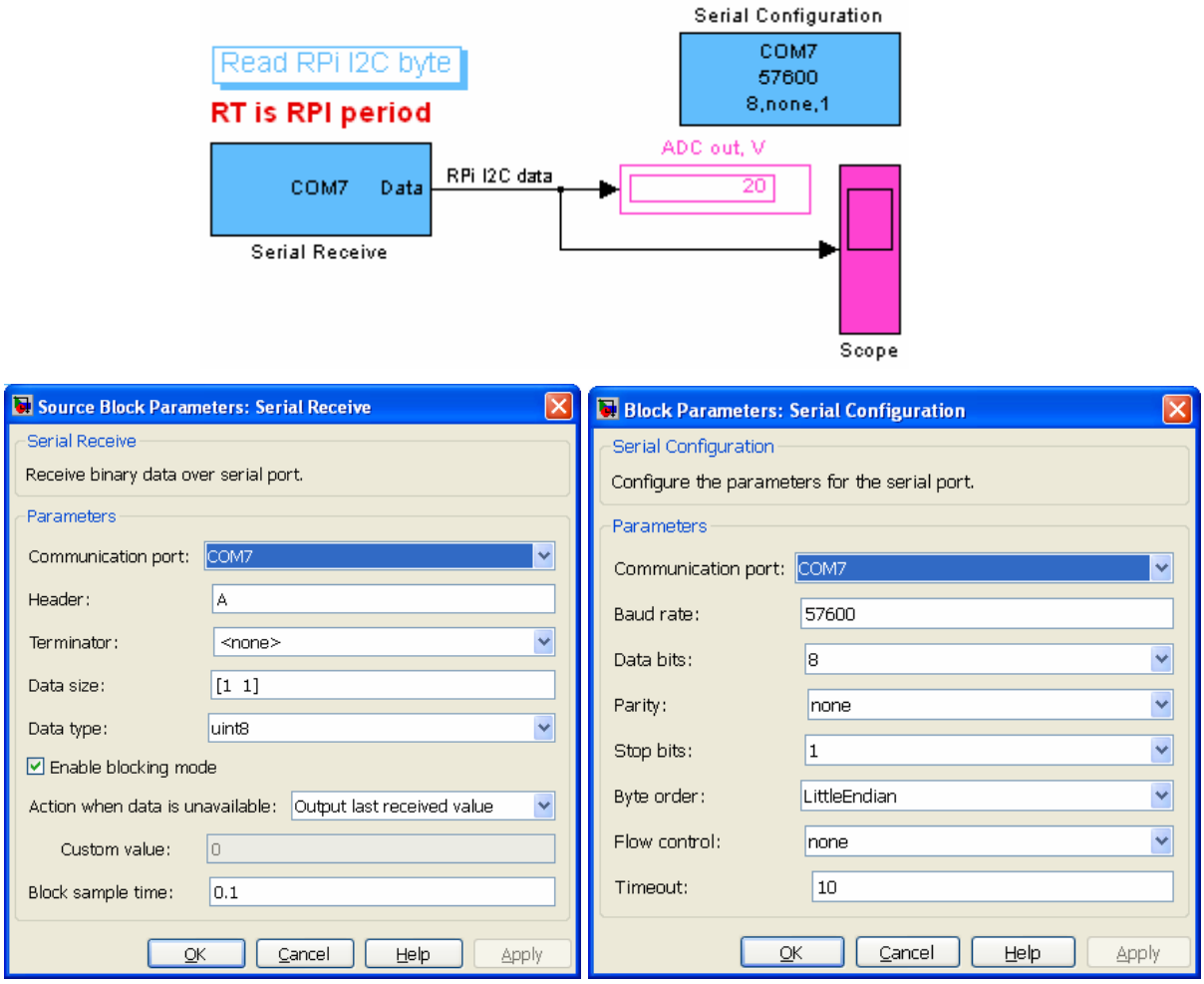


Рис. 31. Параметры Simulink модели.

23. Запустите циклическую передачу данных в I2C канал Raspberry Pi.
24. Наблюдая данные отображаемые Simulink моделью убедитесь в работоспособности канала Raspberry Pi – I2C – Arduino – Simulink.
25. Работу Raspberry Pi завершите командой

    $ sudo shutdown –h now

Задание 6. Работа с файловой системой Raspberry Pi с удаленного компьютера.

1. Загрузите и распакуйте программу WinSSHD для удаленного доступа к файловой системе Raspberry.

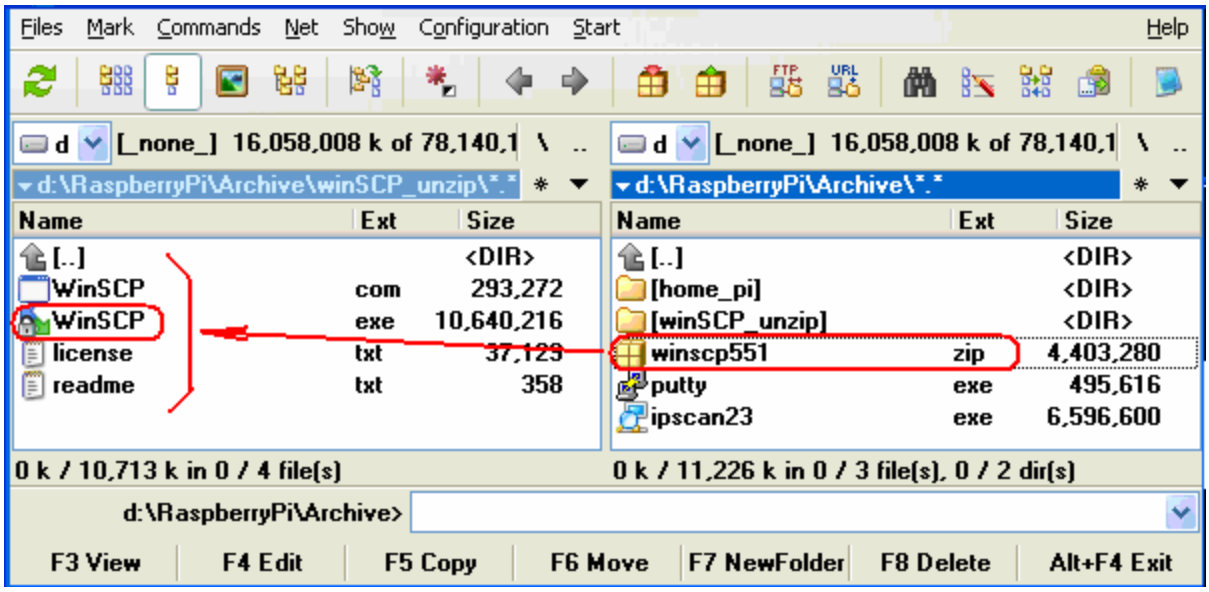


Рис. 32. Распаковка WinSCP.

2. Запустите программу WinSSHD и введите IP адрес Raspberry Pi, его User Name и Password. Имя и пароль, изначально устанавливаемые с ОС Raspbian: **pi** и **raspberry**. Завершите ввод нажатием Login.

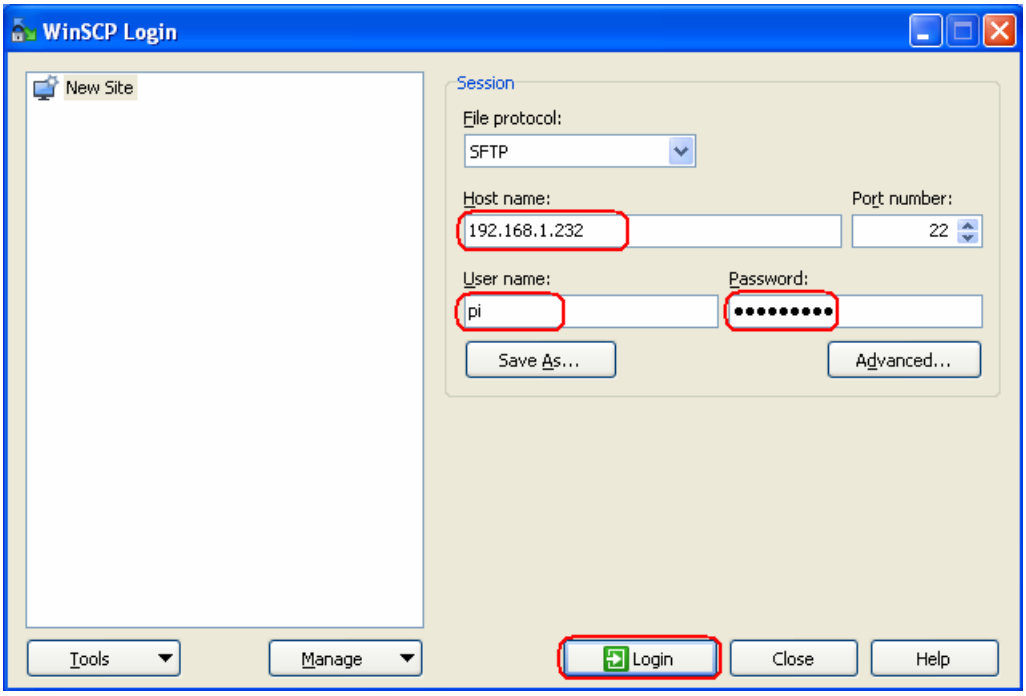


Рис. 33. Ввод параметров компьютера Raspberry Pi.

3. Скопируйте файл или каталог Raspberry Pi на основной компьютер.

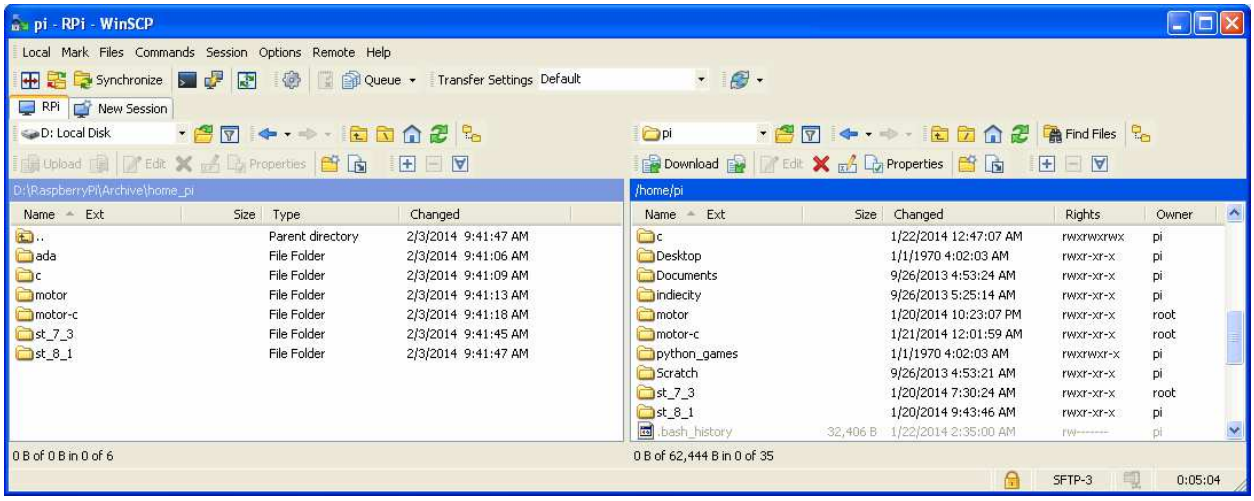


Рис. 34. Оболочка WinSSHD. В левая панель - файловая систем Windows (основной компьютер), правая панель – файловая система Linux (Raspberry Pi).

4. Не разрушая исходное содержимое файловых систем проверьте работу основных команд по работе с файлами (создать, скопировать, раскрыть, переименовать, удалить).

Примечание: Для выполнения ряда команд Linux необходимо иметь права пользователя root.

К Raspberry Pi можно одновременно удаленно обращаться через оболочку WinSSHD и терминальное окно putty, представленное в предыдущем задании.

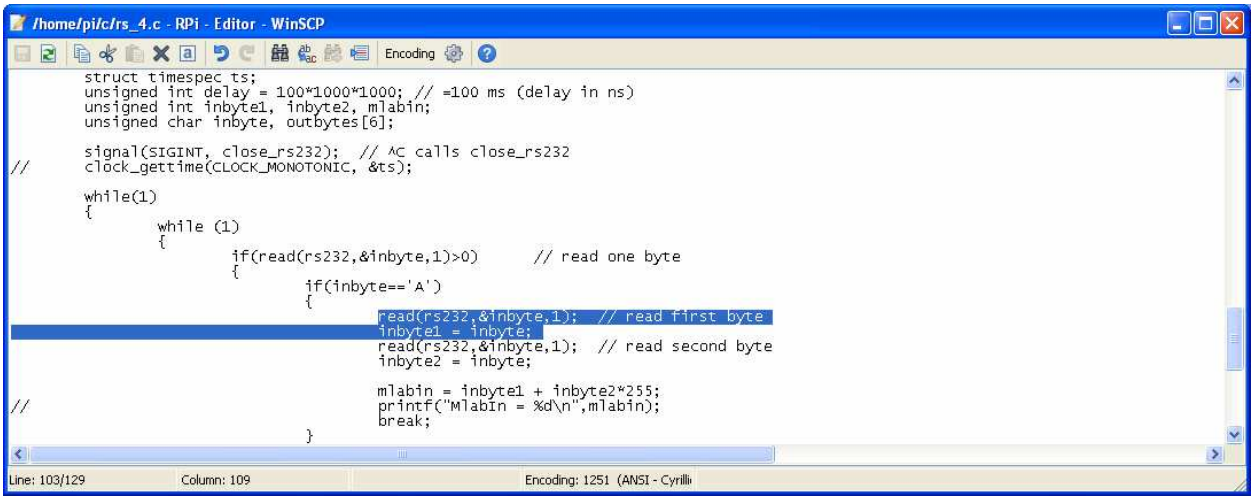
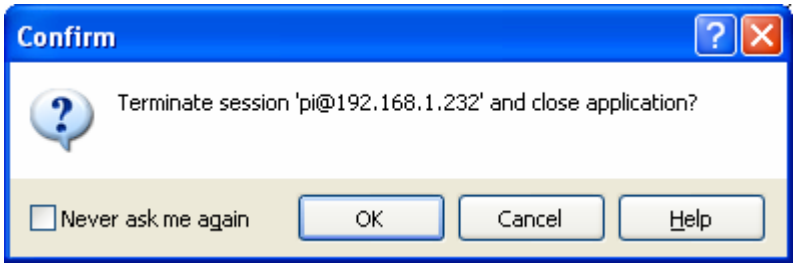


Рис. 35. Окно редактирования файла Raspberry Pi.

5. Закройте WinSSHD.



КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Проведите сравнение компьютера Raspberry Pi с контроллером Arduino.
2. Почему необходим согласователь уровней напряжений при построении канала передачи данных между Raspberry Pi и Arduino?
3. Назовите преимущества канала последовательной передачи данных I2C.
4. Что необходимо учитывать при подключении устройств к Raspberry Pi через RS-232 интерфейс?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Википедия I2C http://ru.wikipedia.org/wiki/I²C
2. Dr. Bob Davidov. Компьютерные технологии управления в технических системах http://portalnp.ru/author/bobdavidov