Task 1:

Solution:

Master:

#include <SPI.h>

void setup (void)

{

digitalWrite(SS, HIGH); // disable Slave Select

SPI.begin ();

SPI.setClockDivider(SPI\_CLOCK\_DIV2);//divide the clock by 8

}

int state;

int oldstate=0;

void loop (void)

{

int c ;

state = !oldstate;

digitalWrite(SS, LOW); // enable Slave Select

// send test string

c = state;

SPI.transfer(c);

digitalWrite(SS, HIGH);

oldstate = state;// disable Slave Select

delay(500);

}

Slave:

#include <SPI.h>

int buff;

volatile boolean process;

void setup (void)

{

pinMode(MISO, OUTPUT); // have to send on master in so it set as output

SPCR |= \_BV(SPE); // turn on SPI in slave mode

process = false;

SPI.attachInterrupt(); // turn on interrupt

pinMode(7,OUTPUT);

}

ISR (SPI\_STC\_vect) // SPI interrupt routine {

{ int c = SPDR;

buff = c;

process = true;// read byte from SPI Data Register

}

void loop (void) {

if (process) {

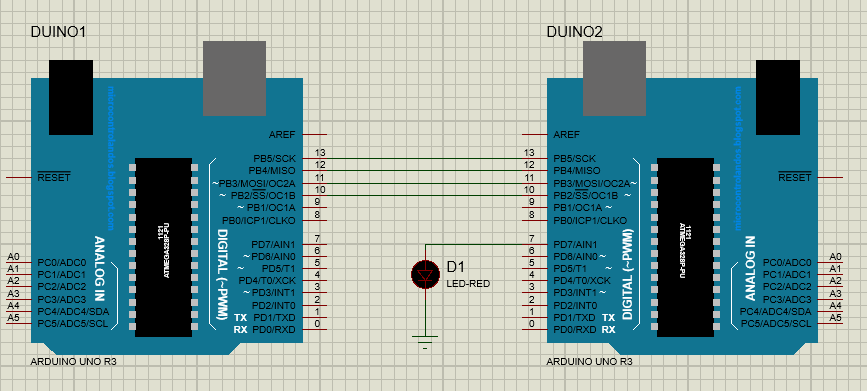
process = false; //reset the process

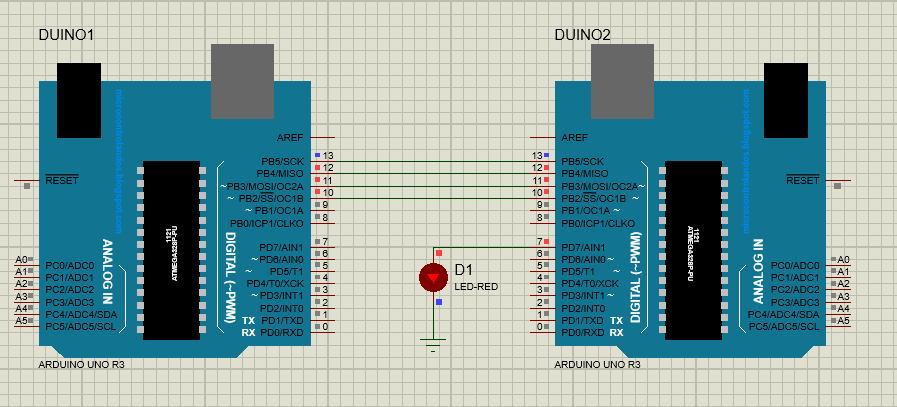
digitalWrite(7,buff);

}

}

Output:





Task 2:

Solution:

Master:

#include <SPI.h>

void setup (void) {

digitalWrite(7,INPUT);

digitalWrite(SS, HIGH); // disable Slave Select

SPI.begin ();

SPI.setClockDivider(SPI\_CLOCK\_DIV2);//divide the clock by 8

}

int state;

int oldstate=0;

void loop (void) {

int c =digitalRead(7);

digitalWrite(SS, LOW); // enable Slave Select

// send test string

SPI.transfer(c);

digitalWrite(SS, HIGH);

}

Slave:

#include <SPI.h>

int buff;

volatile boolean process;

void setup (void) {

pinMode(MISO, OUTPUT); // have to send on master in so it set as output

SPCR |= \_BV(SPE); // turn on SPI in slave mode

process = false;

SPI.attachInterrupt(); // turn on interrupt

pinMode(7,OUTPUT);

}

ISR (SPI\_STC\_vect) // SPI interrupt routine {

{ int c = SPDR;

buff = c;

process = true;// read byte from SPI Data Register

}

void loop (void) {

if (process) {

process = false; //reset the process

digitalWrite(7,buff);

}

}

Output:

