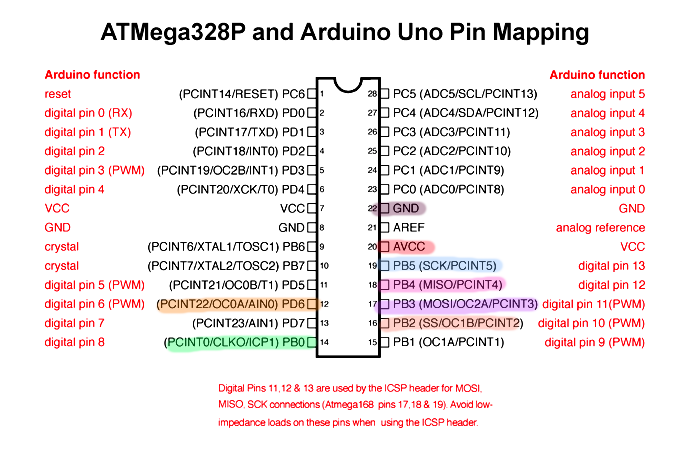
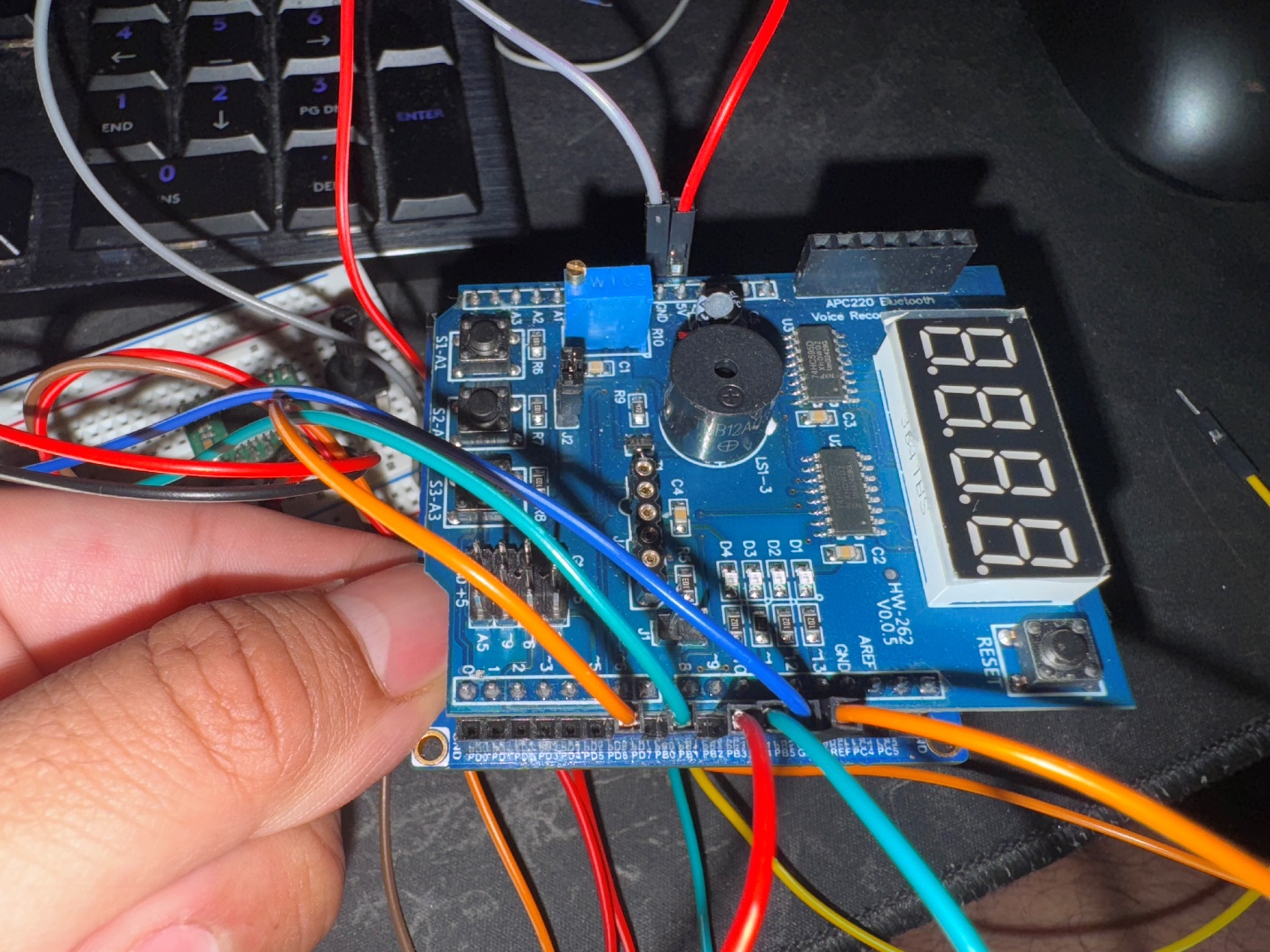
Write, simulate, and demonstrate using Microchip Studio 7 a C code for the AVR ATMEGA328pb microcontroller that performs the following functions:

1. Mount the HC-SR04 Ultrasonic sensor on to the servo motor using the mounting plate/horn. Scan the servo motor from 0 – 120 or 0 -180 deg. Collect the ultrasonic distance (US) distance/raw value continuously during the scan during CW and CCW direction. The resolution of scan must be less than or equal to 2 deg. Display the values in UART as: “Angle, Distance (mm)\n”.

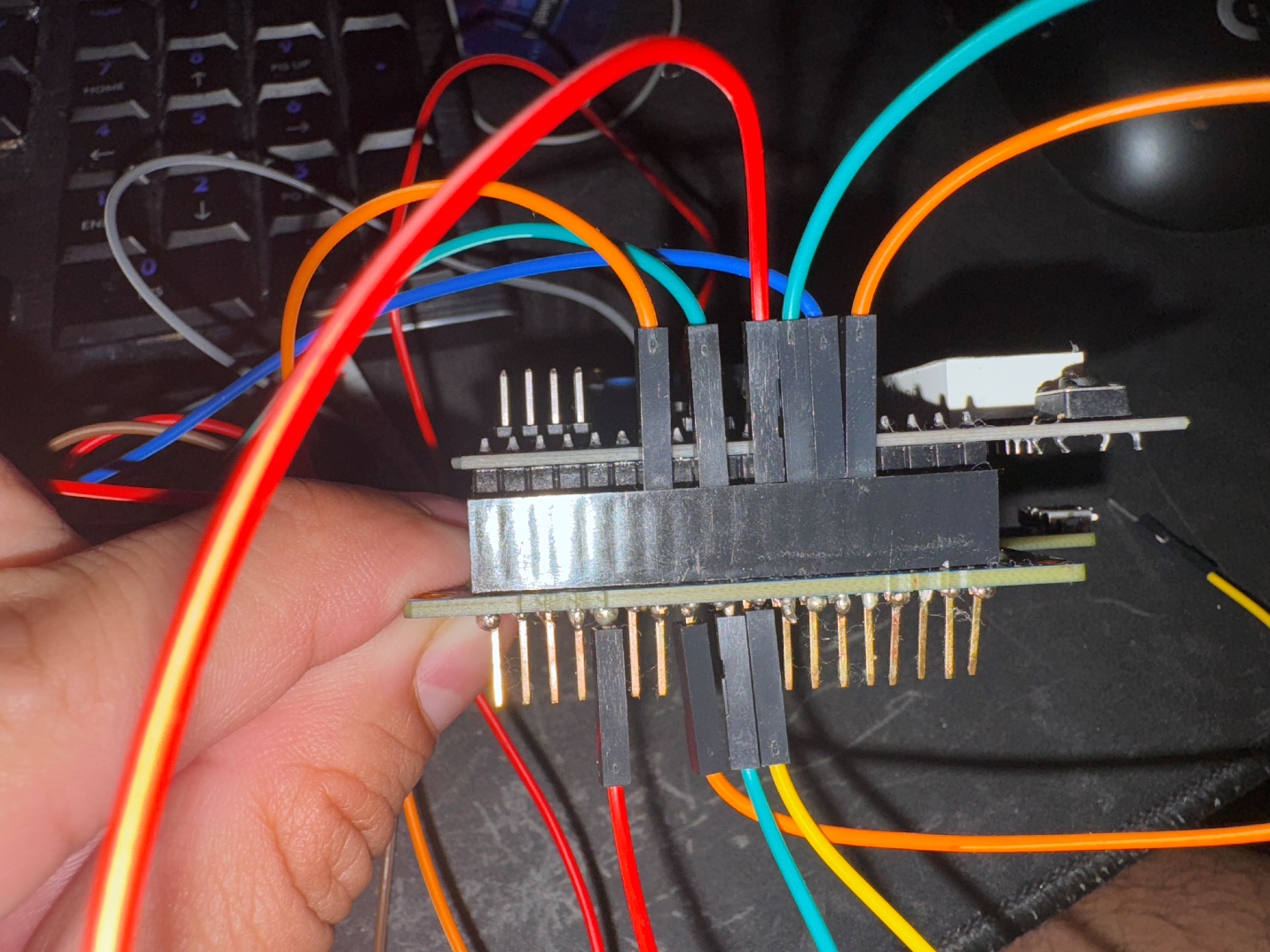
2. Display the value of each scan on the 7-SEG display on the using auto/hardware SPI mode during the CW motion. Display the lowest value of the scan in the 7-SEG display during the CCW arm return time.

**Components Used/Connected**

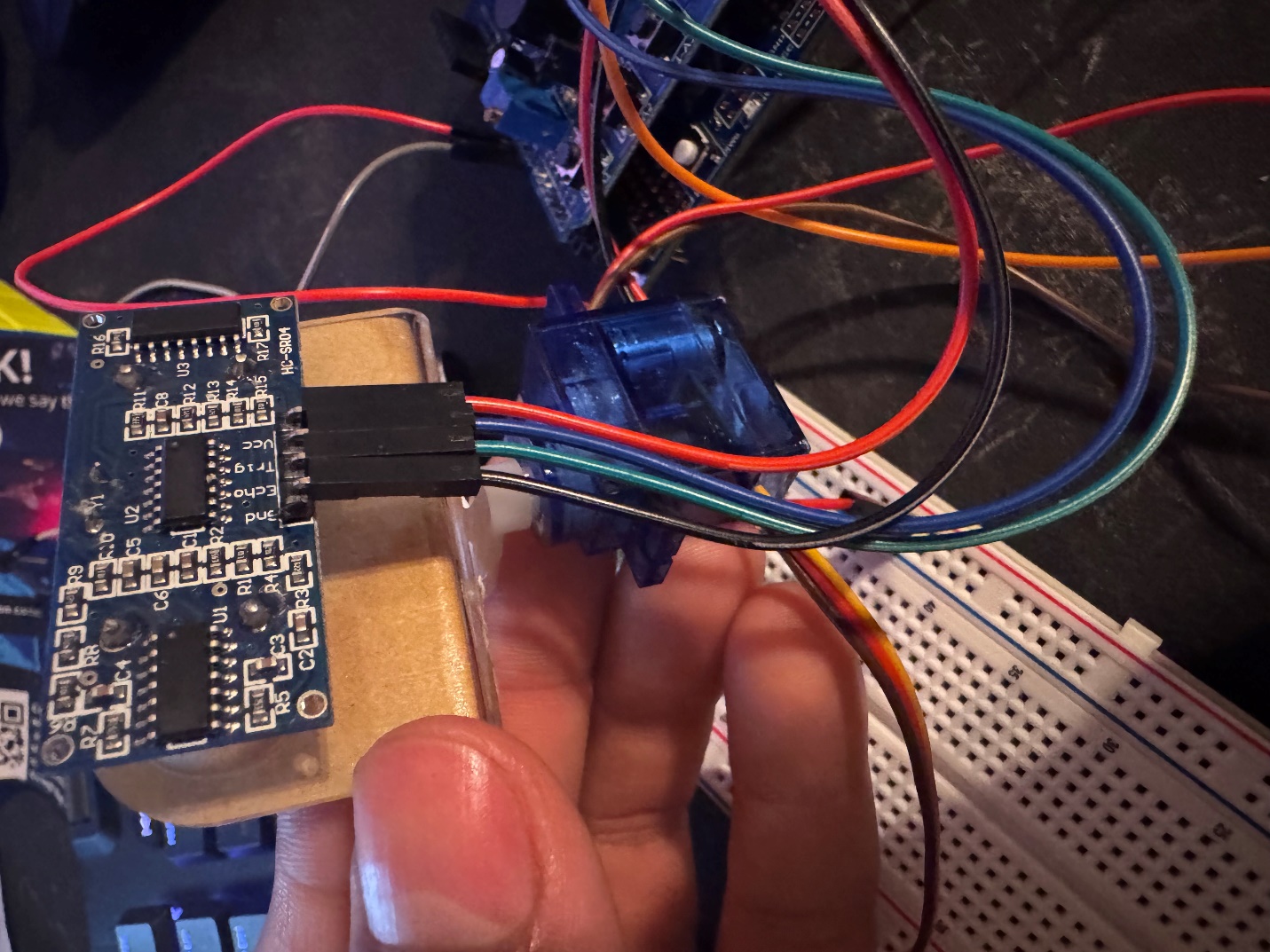




Atmega328p wiring



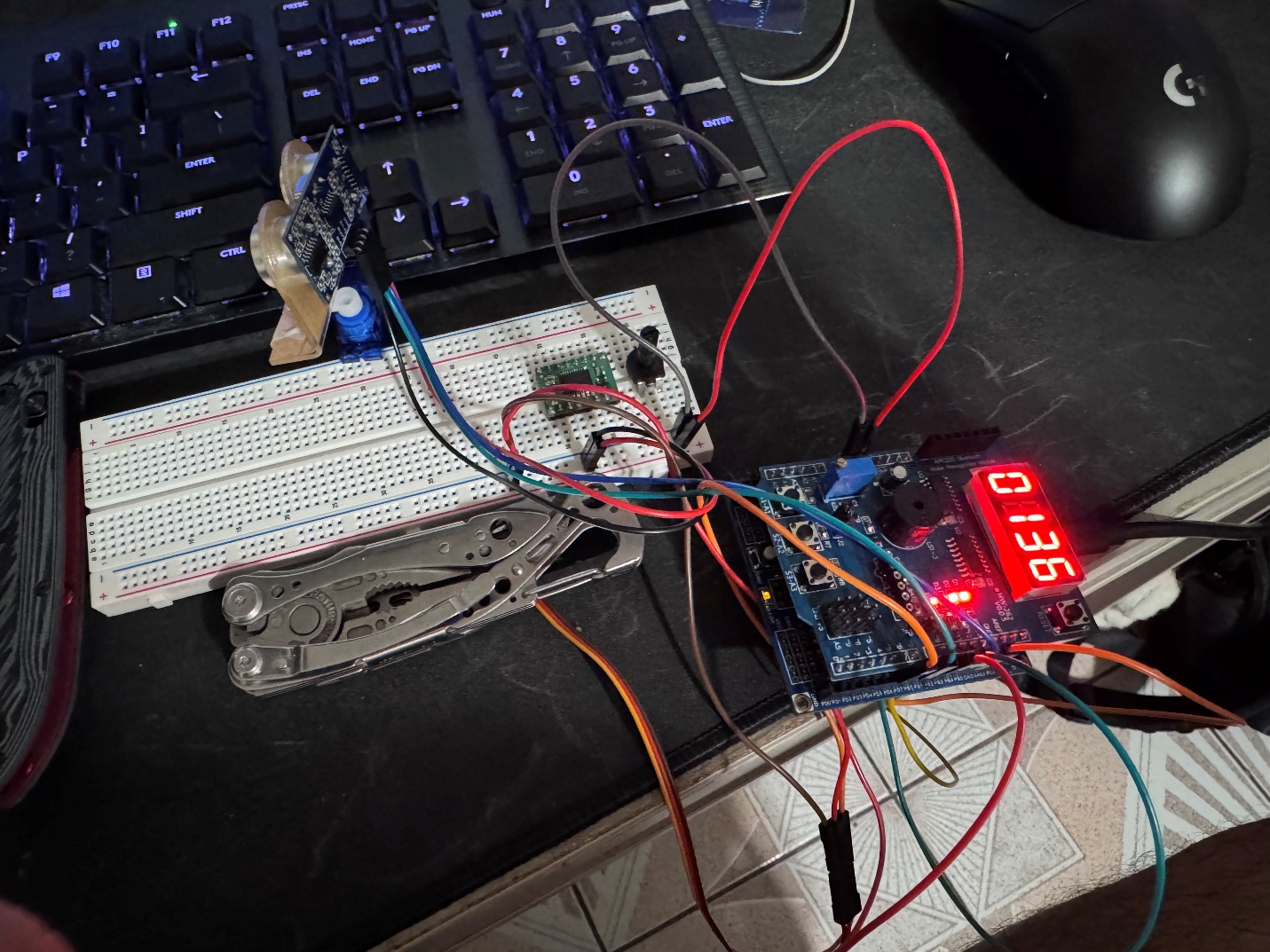
Pins for 7 Segment display (red, orange, green, yellow)



Sensor and Servo motor setup

Sensor (PB0, PB4)

Servo (PD6)



Whole setup.

Had to add weights so the servo could stand upright.

A circuit board with wires and a pair of speakers

AI-generated content may be incorrect.

Schematic showing how the servo and ultrasonic sensor were connected. The 7 Seg was provided through the shield. DATA/MOSI PB3, LATCH/SS PB2, CLOCK/SCK PB5.

**AVR C Code**

/\*

\* DA6.c

\*

\* Created: 5/4/2025 4:48:00 PM

\* Author : enriq

\*/

#define F\_CPU 16000000UL

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#include <stdio.h>

#include <stdint.h>

// 7-segment display

#define DATA PB3

#define CLOCK PB5

#define LATCH PB2

// HC-SR04 | PB4 - TRIG | PB0 - ECHO

#define TRIG\_DDR DDRB

#define TRIG\_PORT PORTB

#define TRIG\_PIN PB4

// Servo motor

#define SERVO\_DDR DDRD

#define SERVO\_PIN PD6 /\* OC0A output \*/

static const uint8\_t SEGMENT\_MAP[10] = {

0xC0,0xF9,0xA4,0xB0,

0x99,0x92,0x82,0xF8,

0x80,0x90

};

static const uint8\_t SELECT\_MAP[4] = {

0xF1,0xF2,0xF4,0xF8

};

volatile uint8\_t disp\_digits[4];

static void shift\_out\_init(void) {

DDRB |= (1<<DATA)|(1<<CLOCK)|(1<<LATCH);

PORTB &= ~((1<<DATA)|(1<<CLOCK)|(1<<LATCH));

}

static void shift\_out(uint8\_t b) {

for(uint8\_t i=0;i<8;i++){

if(b & (1<<(7-i))) PORTB |= (1<<DATA);

else PORTB &= ~(1<<DATA);

PORTB |= (1<<CLOCK);

PORTB &= ~(1<<CLOCK);

}

}

ISR(TIMER2\_OVF\_vect) {

static uint8\_t idx=0;

uint8\_t seg=SEGMENT\_MAP[disp\_digits[idx]];

uint8\_t sel=SELECT\_MAP[idx];

PORTB &= ~(1<<LATCH);

shift\_out(seg);

shift\_out(sel);

PORTB |= (1<<LATCH);

idx = (idx+1)&3;

}

static void tcnt2\_init(void) {

TCCR2A=0;

TCCR2B=(1<<CS21)|(1<<CS20);

TIMSK2=(1<<TOIE2);

}

static void display\_number(int v) {

disp\_digits[0]=(v/1000)%10;

disp\_digits[1]=(v/100)%10;

disp\_digits[2]=(v/10)%10;

disp\_digits[3]=v%10;

}

// UART0 at 9600

static void uart\_init(void) {

UBRR0 = 103; // (16MHz/(16⋅9600))−1

UCSR0B = (1<<TXEN0);

UCSR0C = (1<<UCSZ01)|(1<<UCSZ00);

}

static void uart\_tx(char c) {

while(!(UCSR0A&(1<<UDRE0)));

UDR0=c;

}

static void uart\_print(uint8\_t angle, int mm)

{

char buf[32];

uint8\_t n = snprintf(buf,sizeof(buf),

"%u,%u\n", angle, mm); // “Angle,Distance”

for(uint8\_t i=0;i<n;i++) uart\_tx(buf[i]);

}

static int get\_distance(void) {

// reset timer

TCNT1=0;

TIFR1=(1<<ICF1);

TRIG\_PORT &= ~(1<<TRIG\_PIN);

\_delay\_ms(500);

TRIG\_PORT |= (1<<TRIG\_PIN);

\_delay\_ms(500);

TRIG\_PORT &= ~(1<<TRIG\_PIN);

// rising edge

TCCR1B |= (1<<ICES1);

int t = 30000;

while(!(TIFR1&(1<<ICF1)) && --t);

if(!t) return 0xFFFF;

int start = ICR1;

// falling edge

TIFR1 = (1<<ICF1);

TCCR1B &= ~(1<<ICES1);

t=30000;

while(!(TIFR1&(1<<ICF1)) && --t);

if(!t) return 0xFFFF;

int end = ICR1;

int us = (end - start) / 2;

return ((us \* 171UL) / 10000);

}

static void timer1\_init(void) {

TCCR1A=0;

TCCR1B=(1<<CS11);

}

static void servo\_init(void)

{

SERVO\_DDR |= (1<<SERVO\_PIN);

TCCR0A = (1<<COM0A1) | (1<<WGM00);

TCCR0B = (1<<CS02) | (1<<CS00);

OCR0A = 23;

}

static inline void servo\_set\_angle(uint8\_t deg)

{

if (deg > 180) deg = 180;

OCR0A = 8 + ((int)deg \* 8) / 180;

}

int main(void) {

// disable ADC & comparator

ADCSRA=0;

ACSR =(1<<ACD);

DIDR0 =0x3F;

// TRIG=output, ECHO=input

TRIG\_DDR |= (1<<TRIG\_PIN);

DDRB &= ~(1<<PB0);

shift\_out\_init();

tcnt2\_init();

uart\_init();

timer1\_init();

servo\_init();

sei();

// clear display

display\_number(0);

while(1) {

static uint8\_t angle = 0;

static int8\_t dir = 1; // +1 CW, –1 CCW

servo\_set\_angle(angle);

int distance = get\_distance();

uart\_print(angle, distance);

if (distance != 0xFFFF) // update 7‑seg only on a good echo

display\_number(distance);

angle += dir \* 6;

if (angle >= 180) { angle = 180; dir = -1; }

else if (angle == 0) { dir = 1; }

\_delay\_ms(50);

}

}

A screenshot of a computer program

AI-generated content may be incorrect.

Successful Compilation

A screen shot of a computer

AI-generated content may be incorrect.

Successfully reading values from servo and sensor.

The first value is the angle of servo motor. Second value is sensor data in cm.