**1.0 Prototype**



**2.0 Direct Drive Code**

// PROJECT  :Stepper28BYJ48

// PURPOSE  :Support for demonstration to ICS3U class

// COURSE   :ICS3U

// AUTHOR   :C. D'Arcy

// DATE     :Updated 2024 05 16. 2020 04 28. Confirmed: 2023 05 02.

// MCU      :328P (Nano)

// MOTOR    :MOT-28BYJ-48  (32 teeth, 11.25 degrees between teeth)

// STATUS   :Working

// REFERENCE:https://www.youtube.com/watch?v=B86nqDRskVU

// REFERENCE:https://www.instructables.com/id/BYJ48-Stepper-Motor/

//#include <Stepper.h>        //This libary is OK but (can be) problematic for beginners

//Provide separate 5V supply for motor and time control and motor grounds together

#define IN1 11               //BLUE

#define IN2 10               //PINK

#define IN3 9                //YELLOW

#define IN4 8                //ORANGE

#define TIMEOUT   1000       //control pacing

//------------------------------------------------------------------------------

//For the software-inclined ACES, consider improved code design between barriers

struct driveMode {      //package up the essentials for each drive mode

  uint16\_t increments;

  uint8\_t sequence[8];

  uint8\_t steps;

};

// populate the characteristes of each driving mode

driveMode waveStepping = {32 \* 64, {0, 2, 4, 6, 0, 0, 0, 0}, 4}; //

driveMode fullStepping = {32 \* 64, {1, 3, 5, 7, 0, 0, 0, 0}, 4}; //

driveMode halfStepping = {64 \* 64, {0, 1, 2, 3, 4, 5, 6, 7}, 8}; //

//Activate ONE of the three driving options...

//driveMode active = waveStepping;    //fast, little torque

//driveMode active = fullStepping;    //fast, maximum torque

driveMode active = halfStepping;      //slower, average torque

//------------------------------------------------------------------------------

int8\_t step;                      //drive steps 0-3 or 0-7 (allow for negative)

boolean clockwise = true;         //direction

uint16\_t stepsRemaining = active.increments;  //# of steps for 1 full rotation

void setup() {

  pinMode(IN1, OUTPUT);           //set drive pins for output

  pinMode(IN2, OUTPUT);           //

  pinMode(IN3, OUTPUT);           //

  pinMode(IN4, OUTPUT);           //

  Serial.begin(9600);

  while (!Serial);

  Serial.println(active.increments);

  for (uint8\_t i = 0; i < active.steps; i++)

    Serial.println(active.sequence[i]);

  Serial.println(active.steps);

  Serial.println(stepsRemaining);

}

void loop() {

  //loop through one full shaft rotation at a controlled pace...

  while (stepsRemaining > 0) {

    takeStep();

    delayMicroseconds(TIMEOUT);   // control the pace

    stepsRemaining--;

  }

  delay(TIMEOUT);           //pause before changing direction

  clockwise = !clockwise;   //change direction

  stepsRemaining = active.increments;   //reset the total number of steps for this drive option

}

//Single step through the firing sequence for the active drive option...

//Note1: A HIGH to the ULN2003 pulls the coil end low.

//Note2: This code can be optimized significantly but is shwn in this manner

//       for clarity of interpretation

void takeStep() {

  //be sure to fire the correct sequence for the active drive mode

  switch (active.sequence[step]) {

    case 0:

      digitalWrite(IN1, HIGH);   //BLUE

      digitalWrite(IN2, LOW);    //PINK

      digitalWrite(IN3, LOW);    //YELLOW

      digitalWrite(IN4, LOW);    //ORANGE

      break;

    case 1:

      digitalWrite(IN1, HIGH);

      digitalWrite(IN2, HIGH);

      digitalWrite(IN3, LOW);

      digitalWrite(IN4, LOW);

      break;

    case 2:

      digitalWrite(IN1, LOW);

      digitalWrite(IN2, HIGH);

      digitalWrite(IN3, LOW);

      digitalWrite(IN4, LOW);

      break;

    case 3:

      digitalWrite(IN1, LOW);

      digitalWrite(IN2, HIGH);

      digitalWrite(IN3, HIGH);

      digitalWrite(IN4, LOW);

      break;

    case 4:

      digitalWrite(IN1, LOW);

      digitalWrite(IN2, LOW);

      digitalWrite(IN3, HIGH);

      digitalWrite(IN4, LOW);

      break;

    case 5:

      digitalWrite(IN1, LOW);

      digitalWrite(IN2, LOW);

      digitalWrite(IN3, HIGH);

      digitalWrite(IN4, HIGH);

      break;

    case 6:

      digitalWrite(IN1, LOW);

      digitalWrite(IN2, LOW);

      digitalWrite(IN3, LOW);

      digitalWrite(IN4, HIGH);

      break;

    case 7:

      digitalWrite(IN1, HIGH);

      digitalWrite(IN2, LOW);

      digitalWrite(IN3, LOW);

      digitalWrite(IN4, HIGH);

      break;

    default:

      digitalWrite(IN1, LOW);

      digitalWrite(IN2, LOW);

      digitalWrite(IN3, LOW);

      digitalWrite(IN4, LOW);

      //   break;

  }

  nextStep();

}

void nextStep() {

  // determine the next step...

  step = clockwise ? step + 1 : step - 1;

  // check and adjust for wraparound...

  if (step == active.steps)

    step = 0;

  else if (step < 0)

    step = active.steps - 1;

  Serial.println(step);

}