MCT 4334

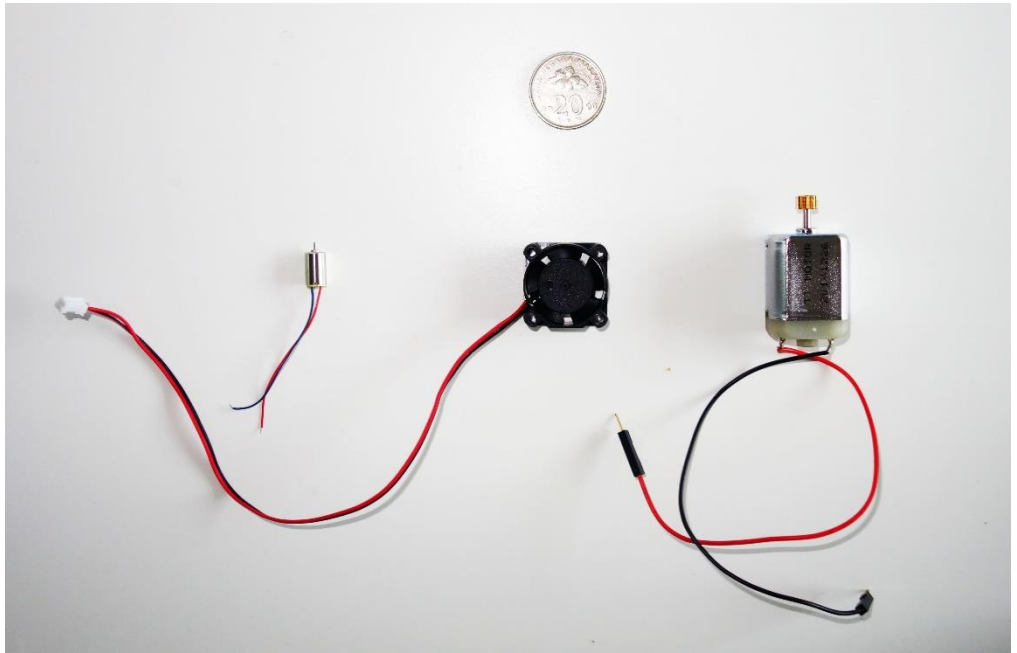
Embedded System Design

Week 10 Interfacing with Motors

Outline

- Controlling DC motors
- Controlling Servo motors
- Controlling Stepper motors

DC motors



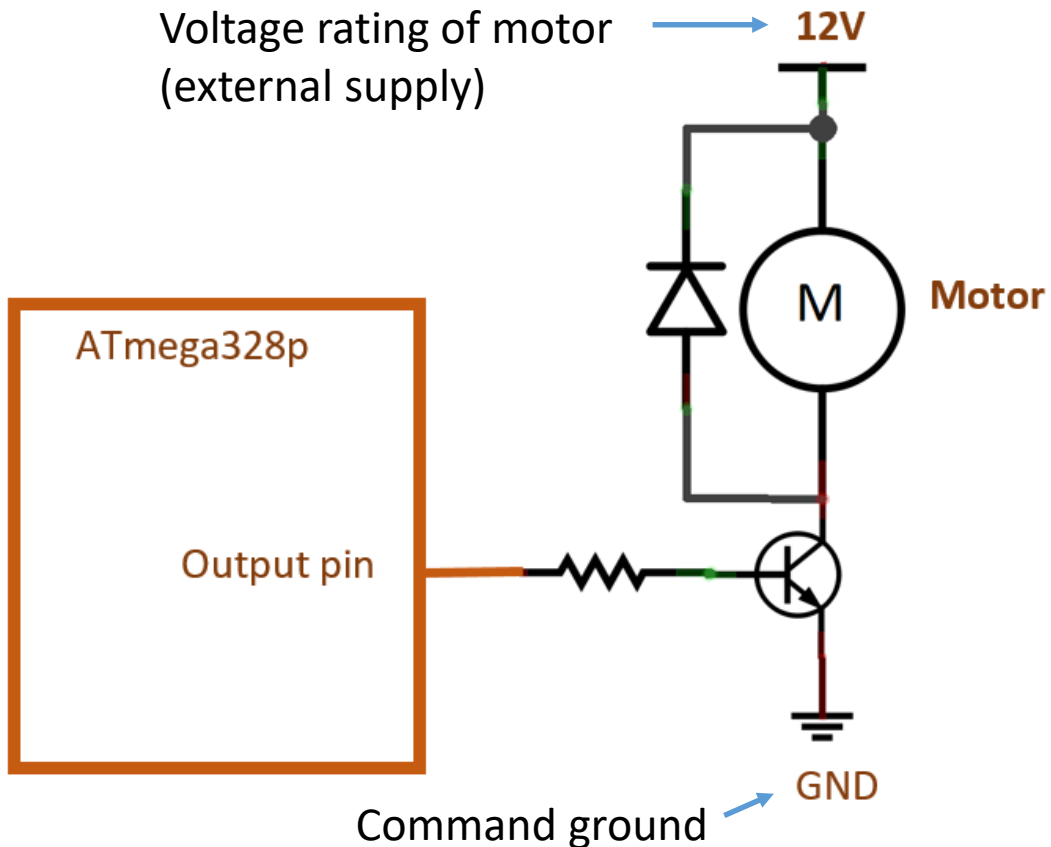
- Usually come in two wires.
- The easiest out of the three motor types to interface with microcontrollers.
- Just apply a voltage across the two wires.
- Reverse the voltage polarity to reverse direction of rotation.

- DC motors come in many sizes
- DC motors have different voltage ratings (6V, 9V, 12V, etc)

- Even tiny DC motors can consume current more than the limit of GPIO pins.
- DC motors should not be connected directly to GPIO pins.

Example of driving a 12V motor

Transistors can be used to interface with motors.



The value of the base resistor should be chosen such that

- when the state of the output pin is LOW, the transistor is in the **cut-off region.** (Collector to emitter is an open circuit)
- when the state of the output pin is HIGH, the transistor is in the **saturation region**. (Collector to emitter is shorted)

- Set the output pin to turn on the motor
- Clear the output pin to turn off the motor

The flyback diode is used to eliminate flyback

How to control the speed of the motor?

Pulse-width modulation (PWM)

Achievable by either

- **Hardware PWM** (on pins with timer outputs) or
- **Software PWM** (can be done any GPIP pins)
  (Just like blinking an LED)

Blinky class revisited

```
class Blinky
{
    private:
      unsigned long OnTime;           //The ON duration in µs
      unsigned long OffTime;          //The OFF duration in µs

      unsigned long last_time=0;      //The last time a transition occurred
      char mask;                      //If pin = 3,  mask    = 0000 1000
      char mask_inv;                  //If pin = 3,  mask_inv= 1111 0111

      char* DDR;
      char* PORT;

   public:
      Blinky()                                          //Constructor without arguments
      {
            Initialize('B', 5, 1000000, 1000000);
      }
      Blinky(char port, int pin, long ontime, long offtime) //Constructor with arguments
      {
            Initialize(port, pin, ontime, offtime);
      }
```

Continued next slide->

```
void Initialize(char port, int pin, long ontime, long offtime)
{
      OnTime = ontime;
      OffTime = offtime;

      switch (port)
      {
        case 'B':
          DDR = (char*) 0x24;
          PORT = (char*) 0x25;
          break;
        case 'C':
          DDR = (char*) 0x27;
          PORT = (char*) 0x28;
          break;
        case 'D':
          DDR = (char*) 0x2A;
          PORT = (char*) 0x2B;
          break;
      }
      mask = (1 << pin);     //The expression (1 << pin) will be used often, it is faster to store it
      mask_inv = ~mask;      //The inverse of the expression (1 << pin) will also be used often
      *DDR |= mask;          //Set the pin as output pin
}
void ChangeOnOffTimes(long ontime, long offtime)
{
      OnTime = ontime;
      OffTime = offtime;
}
```

Continued next slide->

```
    //The refresh function needs to be constantly called. It performs respective transitions.
    void Refresh()
    {
        unsigned long now = micros();

        if (*PORT & mask)                       //The pin is currently HIGH
        {
            if (now - last_time >= OnTime)      //If the pin has been HIGH for long enough
            {
                if (OffTime > 0)                //If OffTime is 0, no need to even turn it OFF
                {
                    *PORT &= mask_inv;          //Make the pin LOW
                }
                last_time = now;                //Take note of the time
            }
        }
        else                                    //The pin is currently LOW
        {
            if (now - last_time >= OffTime)     //If the pin has been LOW for long enough
            {
                if (OnTime > 0)                 //If OnTime is 0, no need to even turn it ON
                {
                    *PORT |= mask;              //Make the pin HIGH
                }
                last_time = now;                //Take note of the time
            }
        }
    }
};
```

Example 1

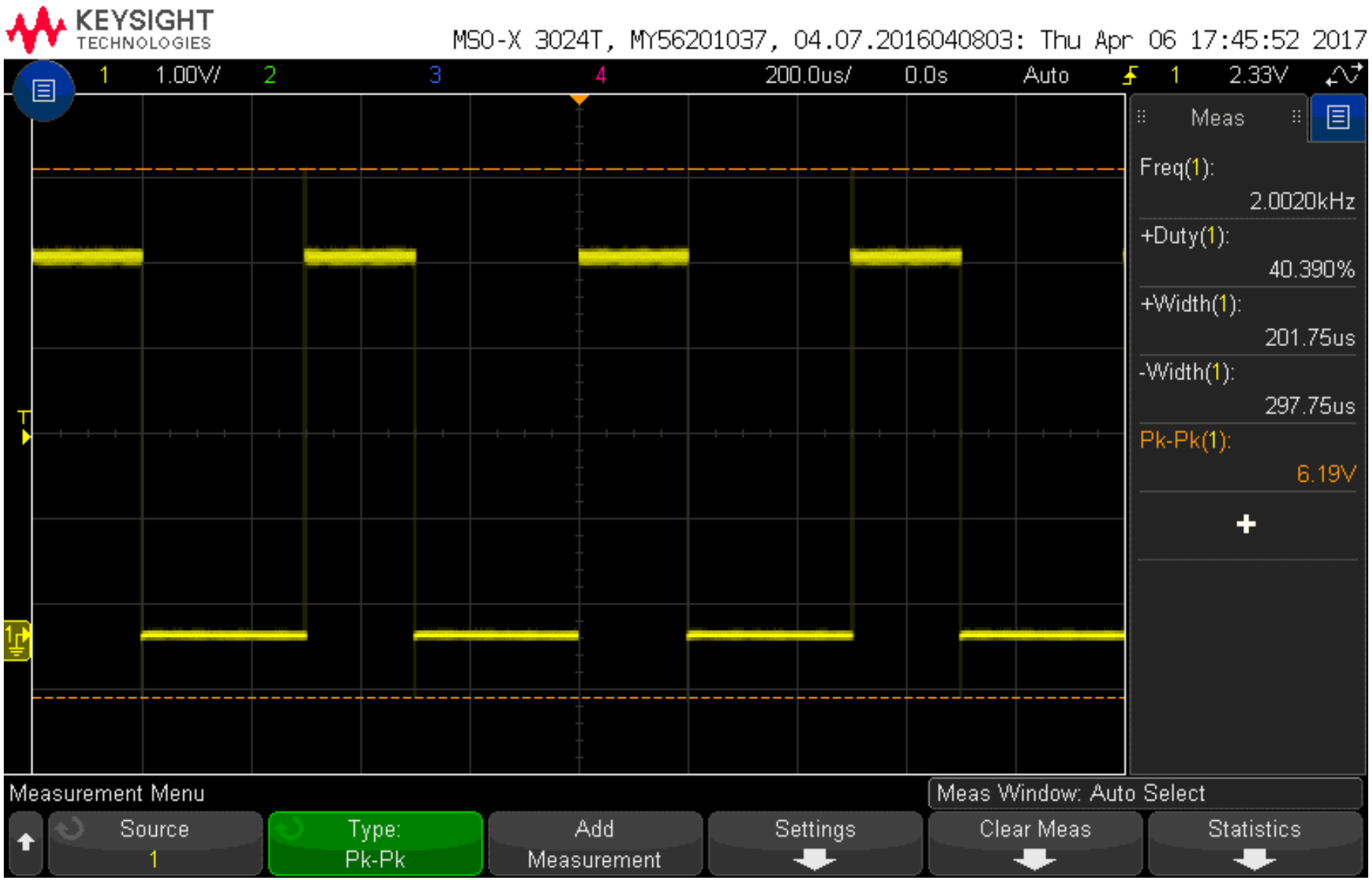
An LED is attached to PB5 of ATmega328p. Use the Blinky class to flash the LED (1s ON and 1s OFF)

```
int main()
{
      Blinky blinker('B', 5, 1000000, 1000000); //Blink PB5  1s ON and 1s OFF

      while(1)
      {
          blinker.Refresh();
      }
}
```

Example 2

Use the Blinky class to generate a PWM waveform at PB5 with on time 200us and off time 300us.

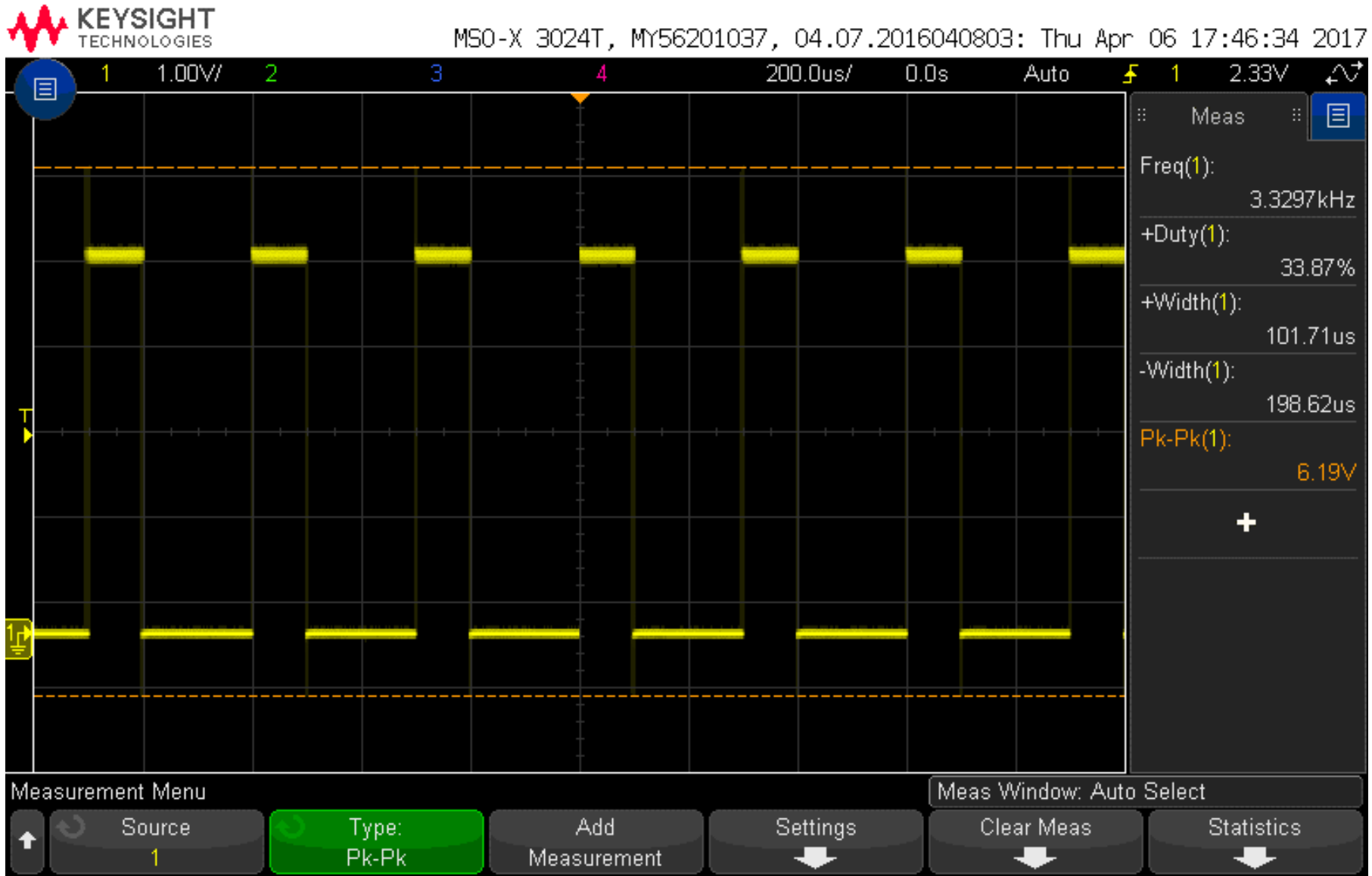
```
int main()
{
      Blinky blinker('B', 5, 200, 300);

      while(1)
      {
          blinker.Refresh();
      }
}
```


KEYSIGHT TECHNOLOGIES
MSO-X 3024T, MY56201037, 04.07.2016040803: Thu Apr 06 17:45:52 2017
1 1.00V/ 2 3 4 200.0us/ 0.0s Auto 1 2.33V
Meas
Freq(1):
2.0020kHz
+Duty(1):
40.390%
+Width(1):
201.75us
-Width(1):
297.75us
Pk-Pk(1):
6.19V
Measurement Menu
Meas Window: Auto Select
Source 1
Type: Pk-Pk
Add Measurement
Settings
Clear Meas
Statistics

Example 3

Use the Blinky class to generate a PWM waveform at PB5 with on time 100us and off time 200us.

```
int main()
{
      Blinky blinker(‘D', 3, 100, 200);

      while(1)
      {
          blinker.Refresh();
      }
}
```


KEYSIGHT TECHNOLOGIES
MSO-X 3024T, MY56201037, 04.07.2016040803: Thu Apr 06 17:46:34 2017
1 1.00V/ 2 3 4 200.0us/ 0.0s Auto 1 2.33V
Meas
Freq(1):
3.3297kHz
+Duty(1):
33.87%
+Width(1):
101.71us
-Width(1):
198.62us
Pk-Pk(1):
6.19V
Measurement Menu
Meas Window: Auto Select
Source 1
Type: Pk-Pk
Add Measurement
Settings
Clear Meas
Statistics

A class for DC motors

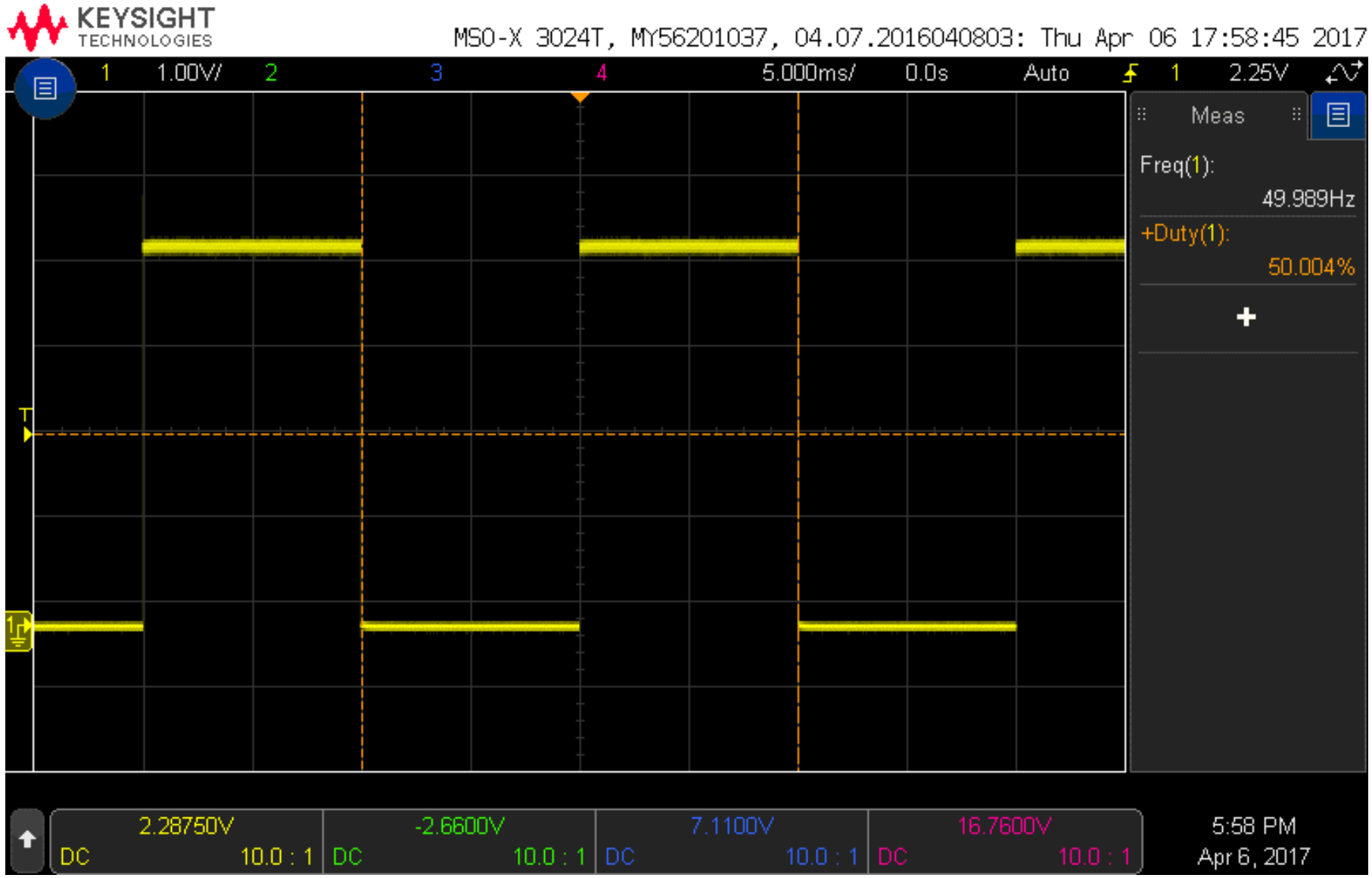
```
class DCMotor
{
      private:
          Blinky oscillator;
          unsigned long Period;

      public:
          DCMotor()
          {
                Initialize('B', 5, 100);
          }
          DCMotor(char port, int pin, double frequency_in_herts)
          {
                Initialize(port, pin, frequency_in_herts);
          }
          void Initialize(char port, int pin, double frequency_in_herts)
          {
                Period = 1000000 / frequency_in_herts;
                oscillator.Initialize(port, pin, 0, Period);
          }
          void Write(double duty_cycle)
          {
                unsigned long OnTime = Period * duty_cycle;
                unsigned long OffTime = Period - OnTime;
                oscillator.ChangeOnOffTimes(OnTime, OffTime);
          }
          void Refresh()
          {
                oscillator.Refresh();
          }
};
```

A DC Motor class can be constructed using the Blinky class

Example 4

Use the DCMotor class to generate a 50Hz PWM waveform at PB0. Keep the duty cycle constant at 50%.

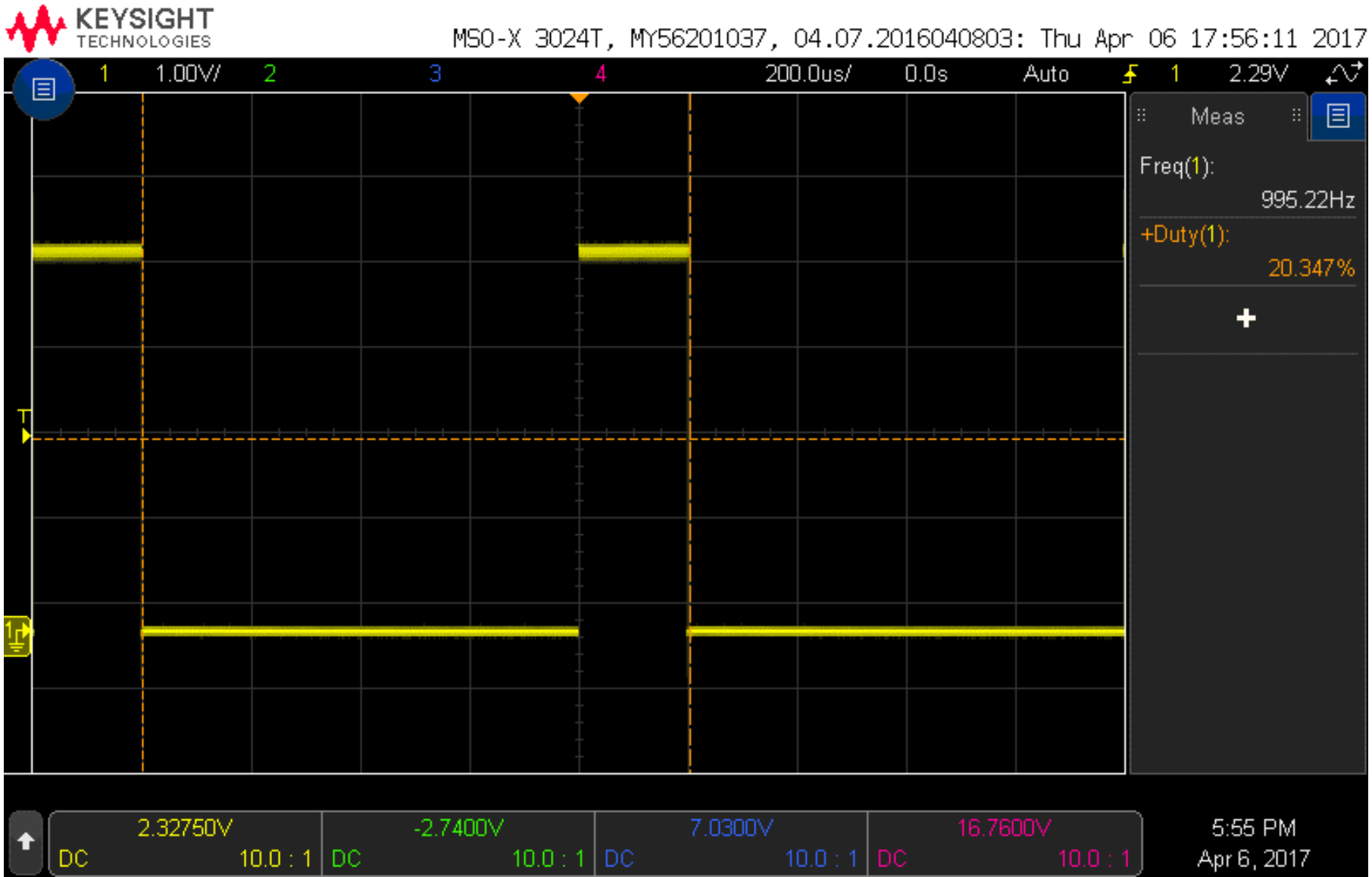
```
int main()
{
	DCMotor motor1('B', 0, 50);
	motor1.Write(0.5);

	while(1)
	{
		motor1.Refresh();
	}
}
```


KEYSIGHT
TECHNOLOGIES
MSO-X 3024T, MY56201037, 04.07.2016040803: Thu Apr 06 17:58:45 2017
1 1.00V/ 2 3 4 5.000ms/ 0.0s Auto 1 2.25V
Meas
Freq(1):
49.989Hz
+Duty(1):
50.004%
2.28750V
DC 10.0 : 1
-2.6600V
DC 10.0 : 1
7.1100V
DC 10.0 : 1
16.7600V
DC 10.0 : 1
5:58 PM
Apr 6, 2017

Example 5

Use the DCMotor class to generate a 1kHz PWM waveform at PB0. Keep the duty cycle constant at 20%.

```
int main()
{
	DCMotor motor1('B', 0, 1000);
	motor1.Write(0.2);

	while(1)
	{
		motor1.Refresh();
	}
}
```

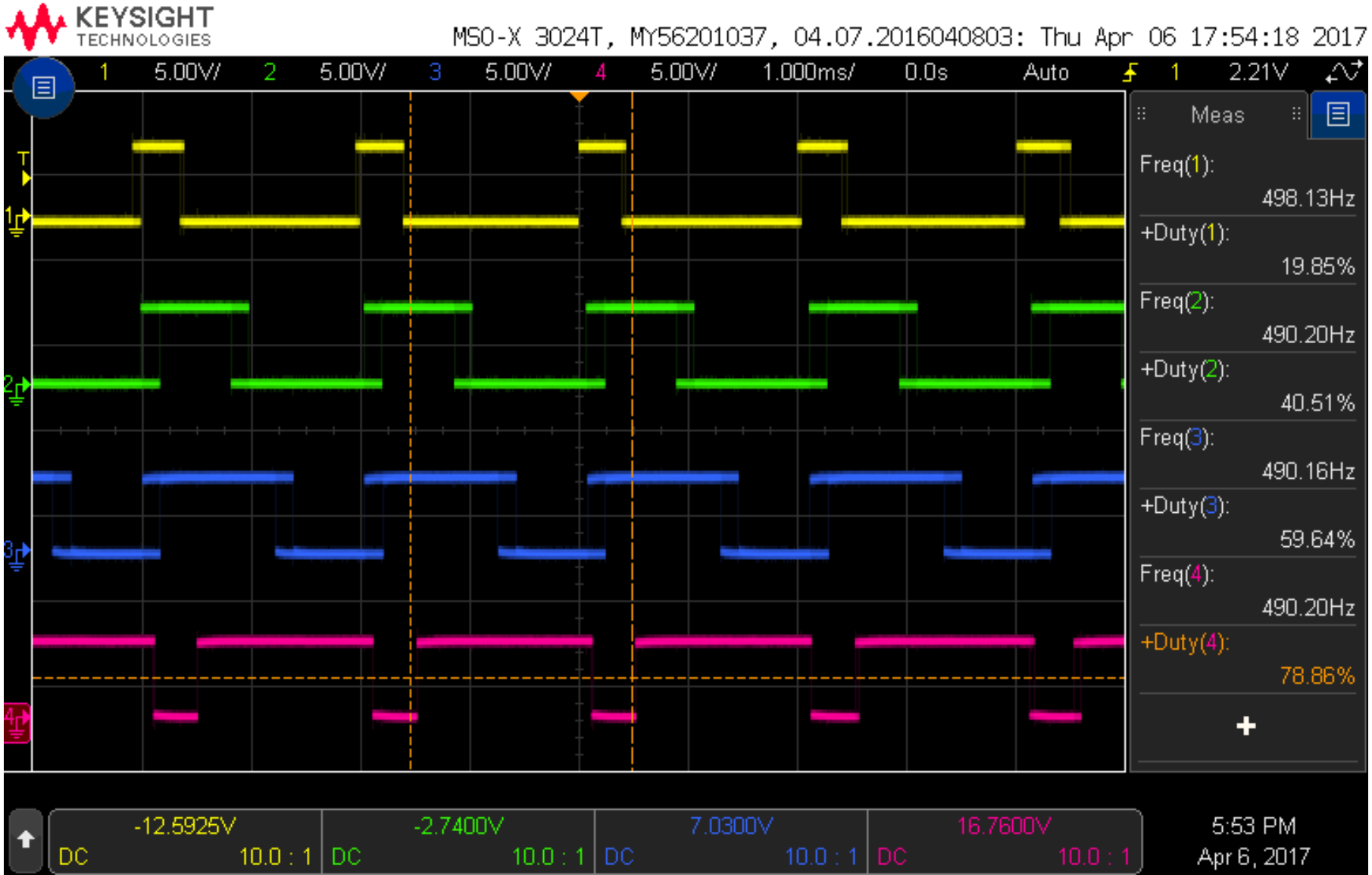

KEYSIGHT TECHNOLOGIES
MSO-X 3024T, MY56201037, 04.07.2016040803: Thu Apr 06 17:56:11 2017
1 1.00V/ 2 3 4 200.0us/ 0.0s Auto 1 2.29V
Meas
Freq(1):
995.22Hz
+Duty(1):
20.347%
2.32750V DC 10.0 : 1
-2.7400V DC 10.0 : 1
7.0300V DC 10.0 : 1
16.7600V DC 10.0 : 1
5:55 PM
Apr 6, 2017

Example 6

```
int main()
{
	DCMotor motor1('B', 0, 500);
	DCMotor motor2('B', 1, 500);
	DCMotor motor3('B', 2, 500);
	DCMotor motor4('B', 3, 500);

	motor1.Write(0.2);
	motor2.Write(0.4);
	motor3.Write(0.6);
	motor4.Write(0.8);

	while(1)
	{
		motor1.Refresh();
		motor2.Refresh();
		motor3.Refresh();
		motor4.Refresh();
	}
}
```

4 DC Motors are connected to PB0, PB1, PB2 and PB3 of ATmega328p via NPN transistors. Use the DCMotor class to drive these motors with PWM frequency of 500Hz with the following duty cycles:

The motor attached to PB0: 20%
The motor attached to PB1: 40%
The motor attached to PB2: 60%
The motor attached to PB3: 80%

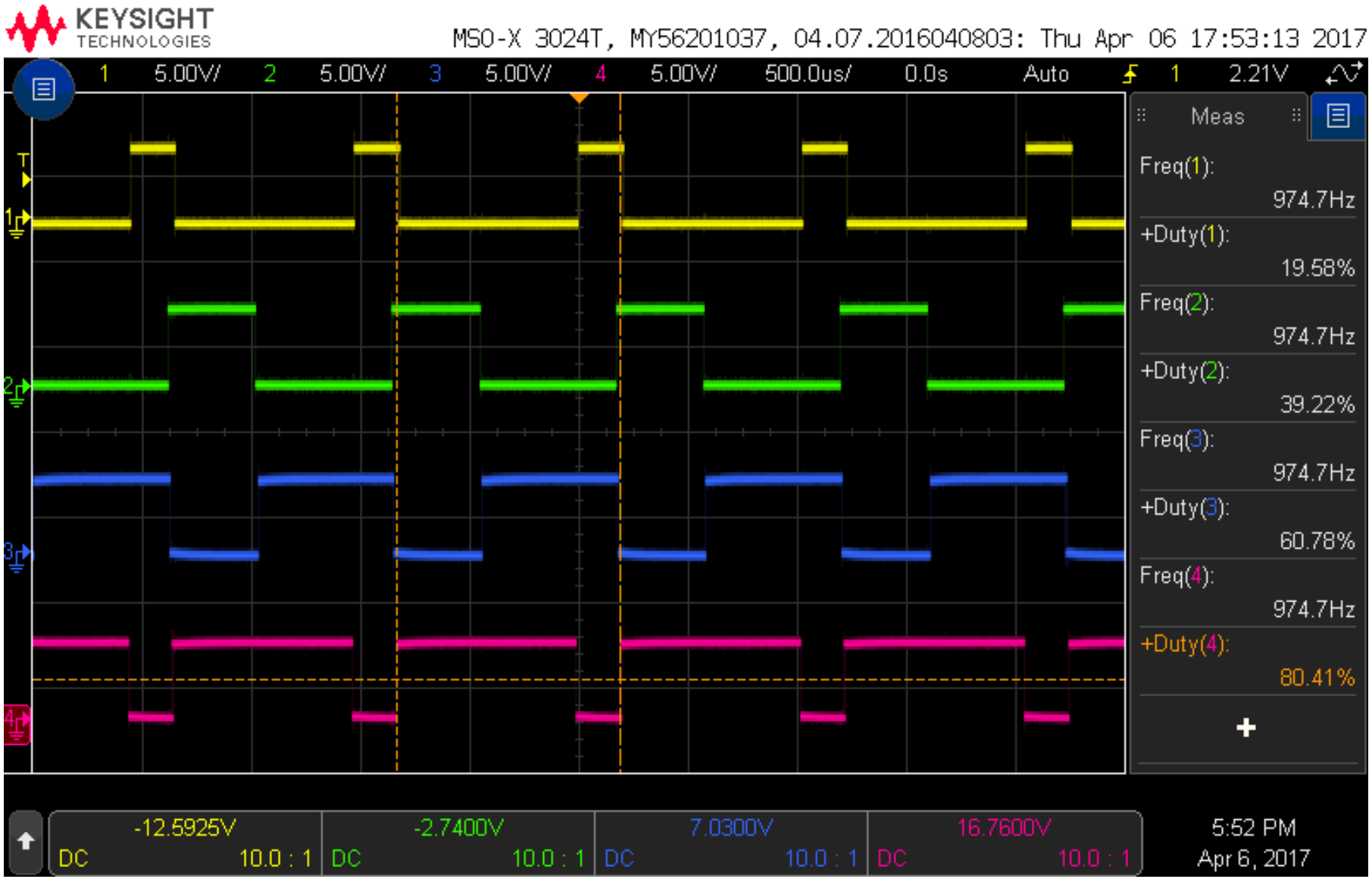

KEYSIGHT
TECHNOLOGIES
MSO-X 3024T, MY56201037, 04.07.2016040803: Thu Apr 06 17:54:18 2017
1 5.00V/ 2 5.00V/ 3 5.00V/ 4 5.00V/ 1.000ms/ 0.0s Auto 1 2.21V
Meas
Freq(1):
498.13Hz
+Duty(1):
19.85%
Freq(2):
490.20Hz
+Duty(2):
40.51%
Freq(3):
490.16Hz
+Duty(3):
59.64%
Freq(4):
490.20Hz
+Duty(4):
78.86%
-12.5925V
DC 10.0 : 1
-2.7400V
DC 10.0 : 1
7.0300V
DC 10.0 : 1
16.7600V
DC 10.0 : 1
5:53 PM
Apr 6, 2017

Example 7

```
int main()
{
	DCMotor motor1('B', 0, 1000);
	DCMotor motor2('B', 1, 1000);
	DCMotor motor3('B', 2, 1000);
	DCMotor motor4('B', 3, 1000);

	motor1.Write(0.2);
	motor2.Write(0.4);
	motor3.Write(0.6);
	motor4.Write(0.8);

	while(1)
	{
		motor1.Refresh();
		motor2.Refresh();
		motor3.Refresh();
		motor4.Refresh();
	}
}
```

4 DC Motors are connected to PB0, PB1, PB2 and PB3 of ATmega328p via NPN transistors. Use the DCMotor class to drive these motors with PWM frequency of 1kHz with the following duty cycles:

The motor attached to PB0: 20%
The motor attached to PB1: 40%
The motor attached to PB2: 60%
The motor attached to PB3: 80%


KEYSIGHT
TECHNOLOGIES
MSO-X 3024T, MY56201037, 04.07.2016040803: Thu Apr 06 17:53:13 2017
1 5.00V/ 2 5.00V/ 3 5.00V/ 4 5.00V/ 500.0us/ 0.0s Auto 1 2.21V
Meas
Freq(1):
974.7Hz
+Duty(1):
19.58%
Freq(2):
974.7Hz
+Duty(2):
39.22%
Freq(3):
974.7Hz
+Duty(3):
60.78%
Freq(4):
974.7Hz
+Duty(4):
80.41%
-12.5925V
DC 10.0 : 1
-2.7400V
DC 10.0 : 1
7.0300V
DC 10.0 : 1
16.7600V
DC 10.0 : 1
5:52 PM
Apr 6, 2017

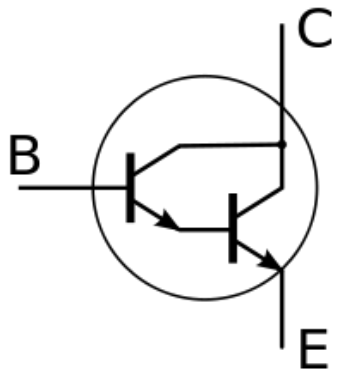
Example 8 (Digital Control System)

The speed of a DC motor can be continuously adjusted as part of a control system.
Assume that PID class and Sensor class exist.

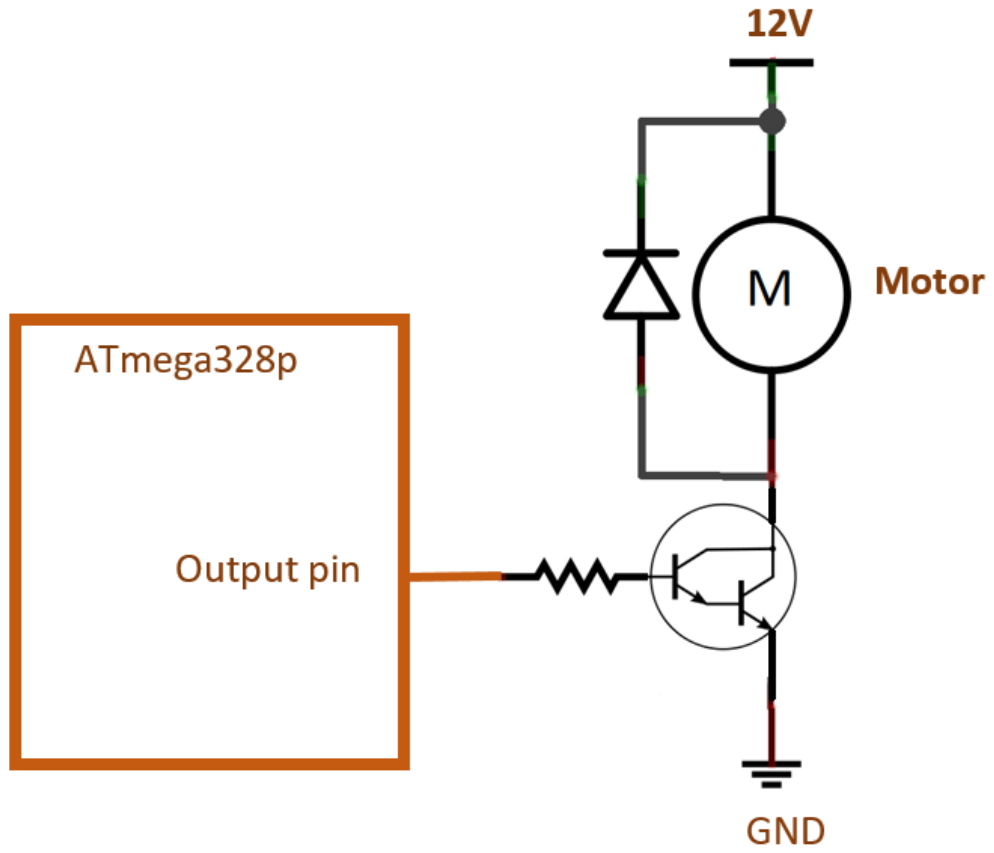
```
int main()
{
	DCMotor motor('B', 0, 1000);//DC Motor attached to PB0
	PID controlsystem;
	Sensor temperature;

	while(1)
	{
		double input = temperture.Acquire();    //Read the value of the temp sensor
		double motor_speed = controlsystem.GetResponse(input); //Calculate response
		motor1.Write(motor_speed);          //Write the output to the motor
		motor1.Refresh();
	}
}
```

Other ways of driving DC motors

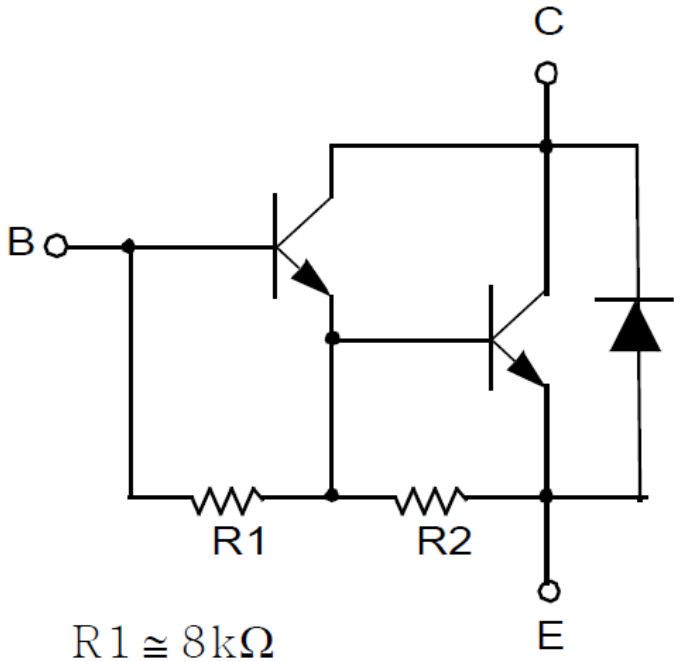
- Darlington transistors can also be used. A Darlington transistor is a pair of NPN transistors cascaded in such a way that the current amplified by the first transistor is further amplified by the second transistor.
- FET transistors can also be used.



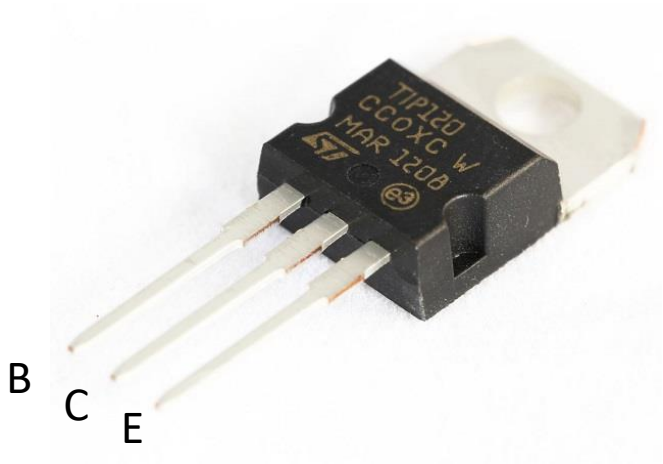
Darlington pair/transistor



- There are many variants of Darlington pairs. TIP120 is a Darlington pair for medium power switching.
- It can withstand collector current up to 5A.



$R1 \cong 8k\Omega$
$R2 \cong 0.12k\Omega$



- ICs that contain an array of Darlington pairs are also available.

- Darlington arrays are often used to drive stepper motors

ULN2003 IC

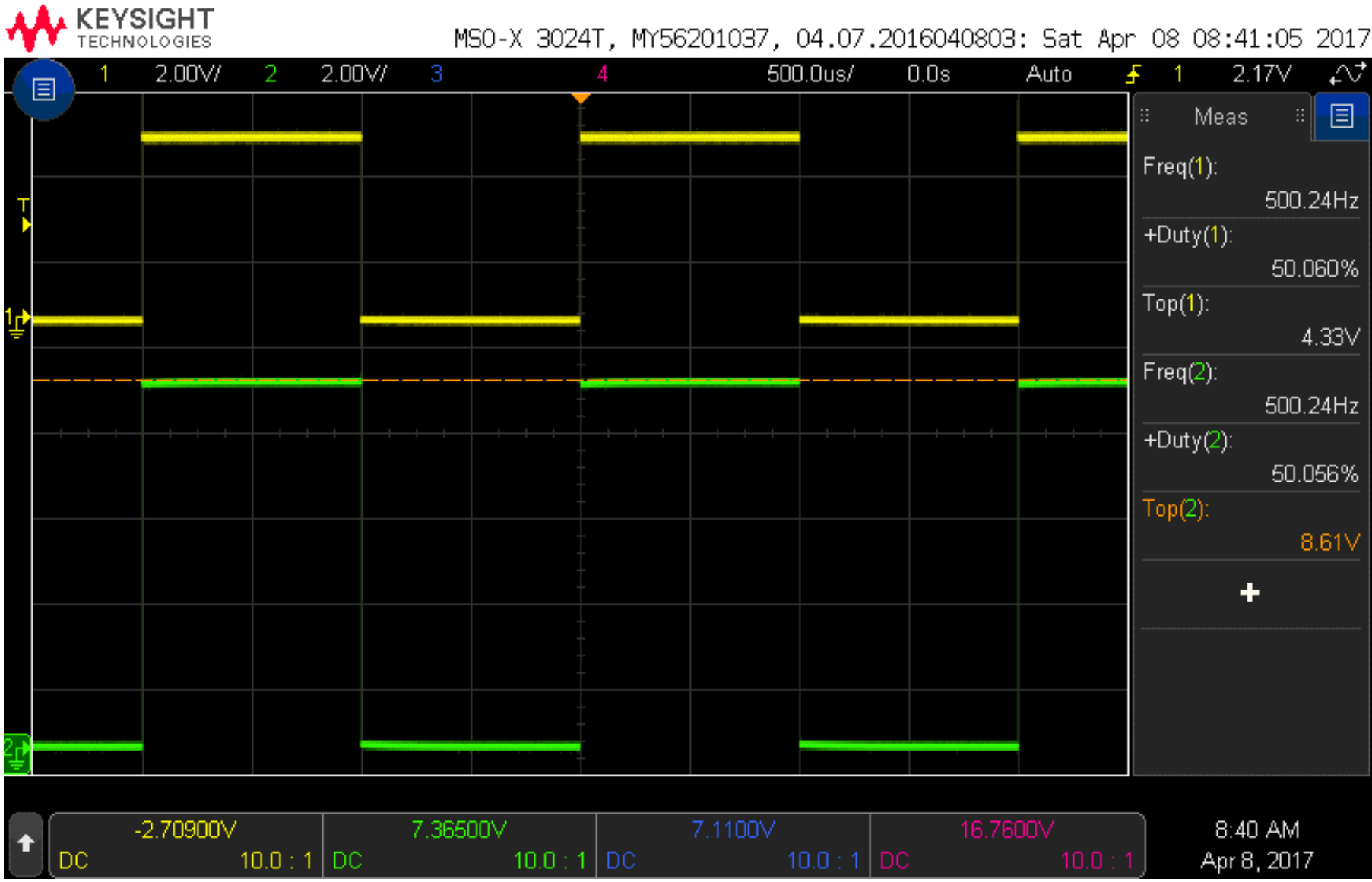
Tips:

- Make sure that the motor does not draw more than collector current rating of the transistor.

- Use appropriate transistors (small signal, medium power, high power)
- Remember that transistors drop some voltage (V_{CE}). The motors will not receive the full voltage.

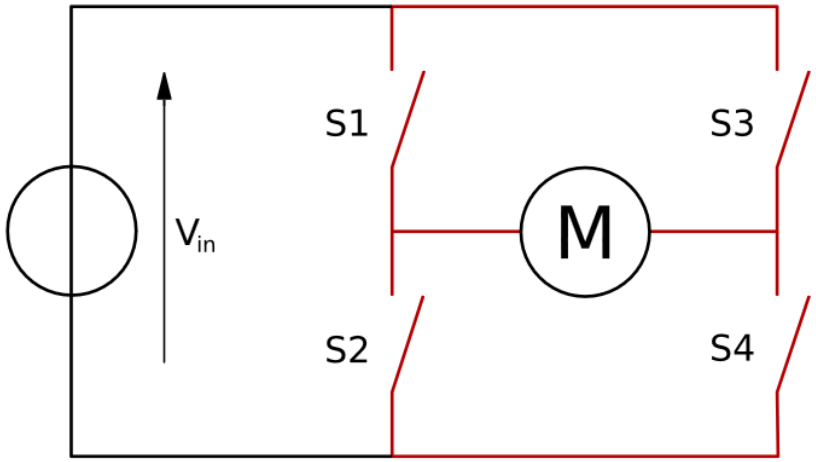
Output of microcontroller

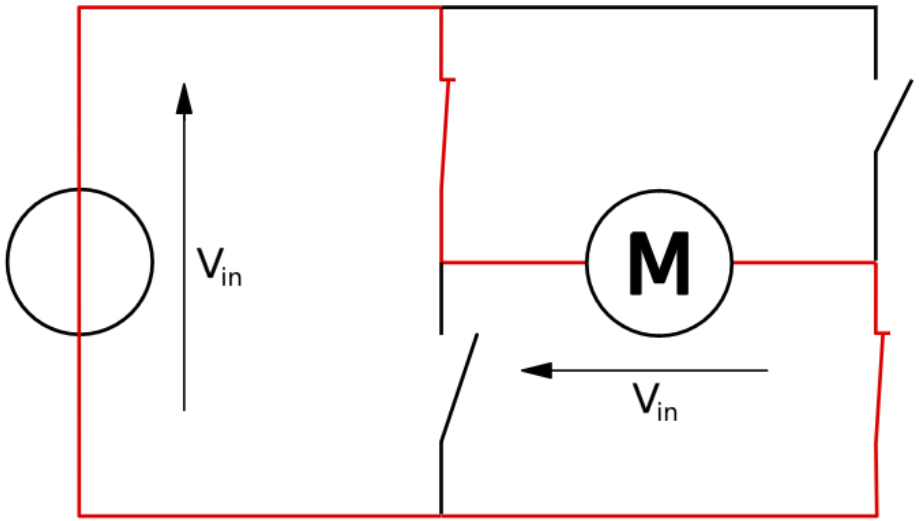
Output of transistor for driving motor

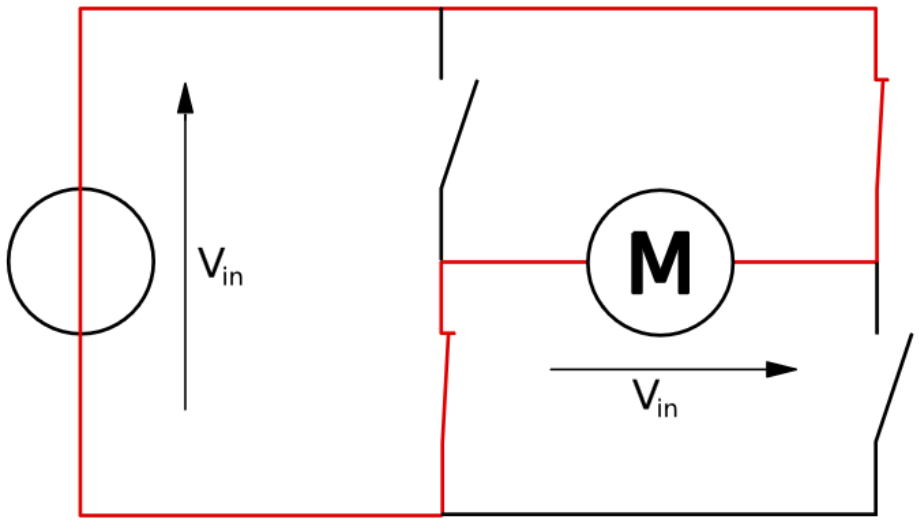


Changing the direction of rotation

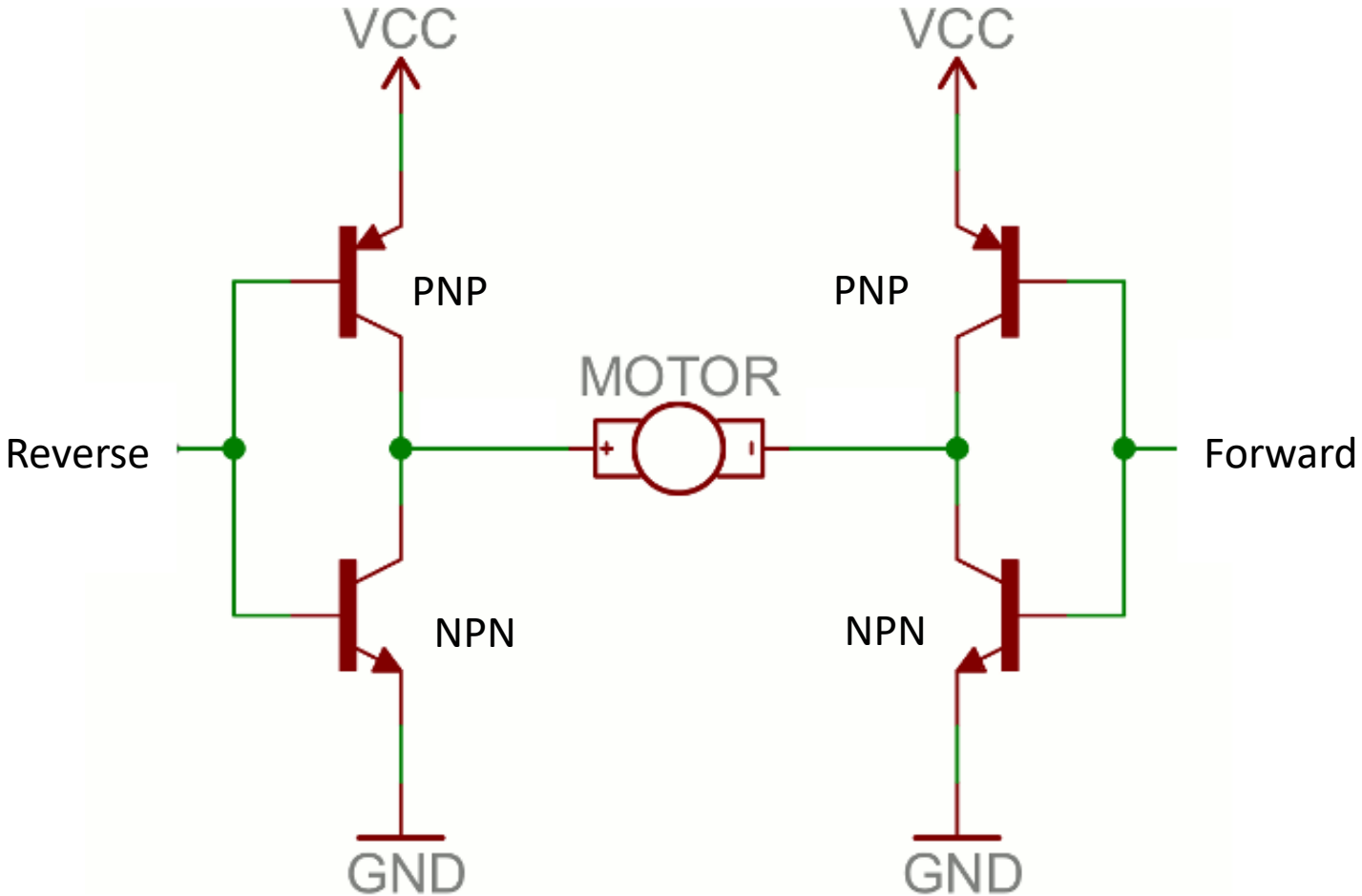
An H-bridge can be used to vary the direction of rotation.







- An H-bridge can be constructed using a pair of SPDT relays or 4 transistors



H bridges are also available in form of ICs

Servo motors

10:10am

- Usually come in three wires (+, GND and signal)
- The signal wire can be connected directly to a GPIO pin (configure that pin as output)
- A typical servo has an onboard controller that reads the signal coming through the signal wire and moves the rotor accordingly.
- Unmodified servos usually rotate up to 180°

Driving Servo Motors

The microcontroller must send pulses to the servo to control the rotation.

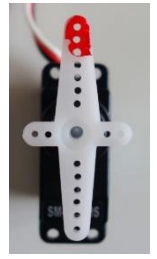
The **width of the pulse** determines the angle of rotation.

Many servos expect to see a pulse every 20ms). But they can also accept 10 to 22ms. (This may differ from servo to servo

Different servos have different minimum pulse width and maximum pulse width. Check the datasheets.

One extreme

Middle



Other extreme

Driving Servo Motors

- The onboard controller of a servo reads the **pulse width** (not the duty cycle)
- A typical servo would move to the **exact position** under the following scenarios:
    - Pulse duration of 1ms every 20ms (Duty cycle 5%)
    - Pulse duration of 1ms every 10ms (Duty cycle 10%)

- Duty cycle does not matter. Pulse width matters!

FAQ on driving servos using hardware PWM

- **Can we drive servos using hardware PWM on ATmega328p?**

  Yes

  But we need to drop the frequency of the PWM signal
  Assuming 16MHz clock speed
  Use 1024 pre-scaler and 8-bit fast PWM
  Frequency of PWM signal = 61.04 Hz (period = 16ms)

- **Is it a good idea?**

  Probably not

  Very poor resolution.
  Suppose min pulse width = 1ms and max pulse width = 2ms.
  We are limited to 6.25% to 12.25% duty cycles (for 0° to 180° rotation of servo)

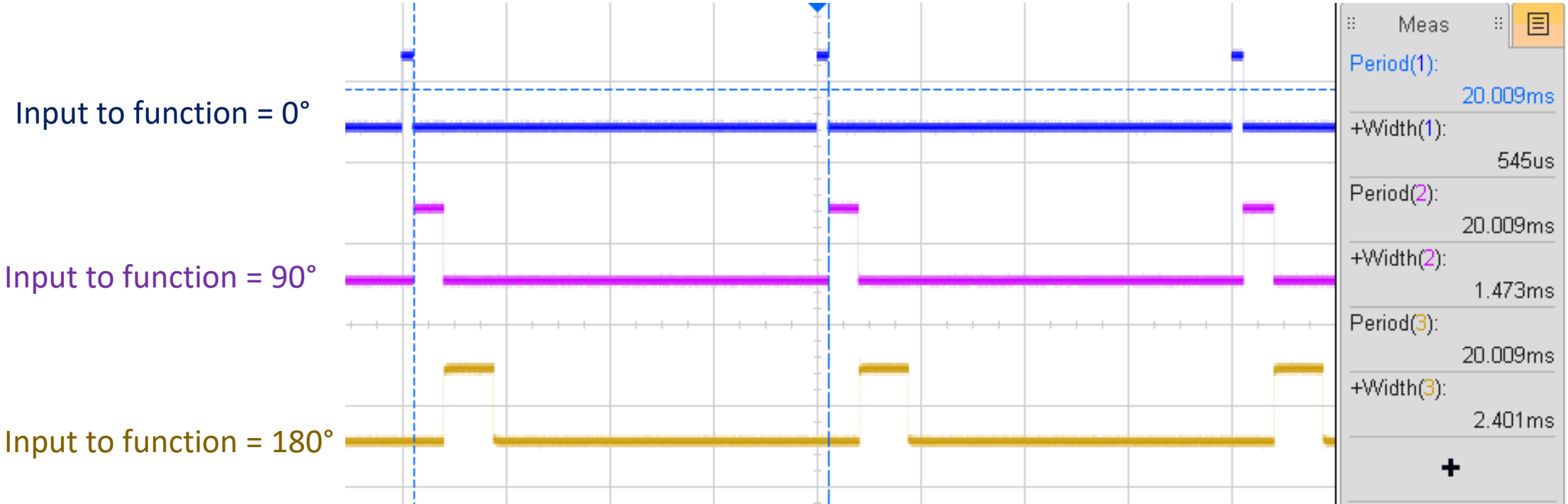
Arduino Servo Library

- Arduino has an OOP-based servo library.
- It contains a function called write() which accepts angle between 0° to 180°.

0° = 0.5440 ms pulse width

180° = 2.400 ms pulse width

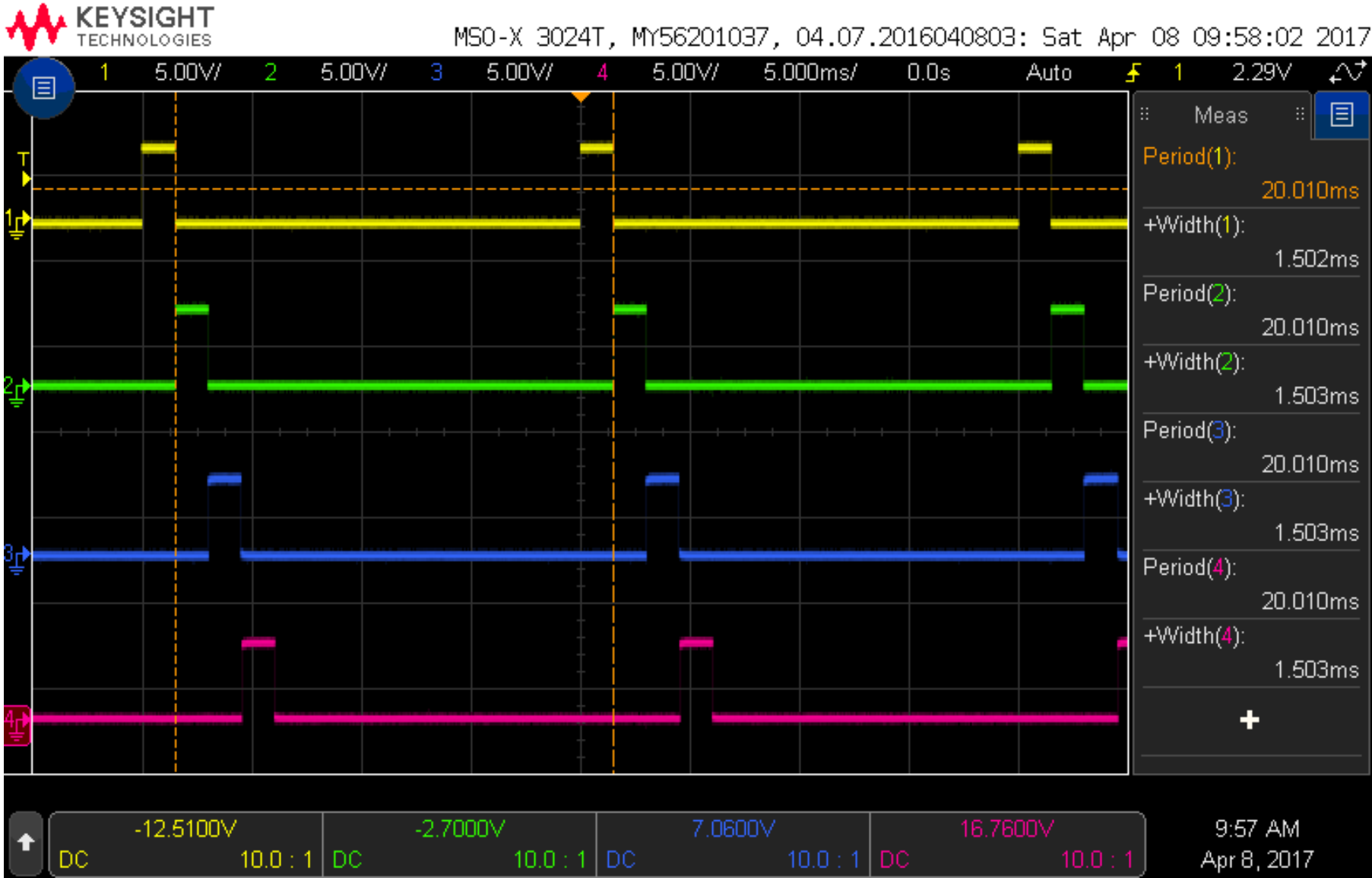
- The Arduino library generated pulses with period of 20 ms.
- The max and min pulse widths can be calibrated



Driving 4 servos at the same time using the Arduino library (90° to all the servos)

The 4 control signals are out of phase.

They are actually being driven one at the time.



A class for servo motors

```
class Servo
{
	private:
		Blinky oscillator;
		int Period;

	public:
		Servo()
		{
			Initialize('B', 5, 20000);
		}
		Servo(char port, int pin, int period)
		{
			Initialize(port, pin, period);
		}
		void Initialize(char port, int pin, int period)
		{
			Period = period;
			oscillator.Initialize(port, pin, 0, period);
		}
		void Write(int ontime)
		{
			oscillator.ChangeOnOffTimes(ontime, Period - ontime);
		}
		void Refresh()
		{
			oscillator.Refresh();
		}
};
```

A Servo class can be constructed using the Blinky class

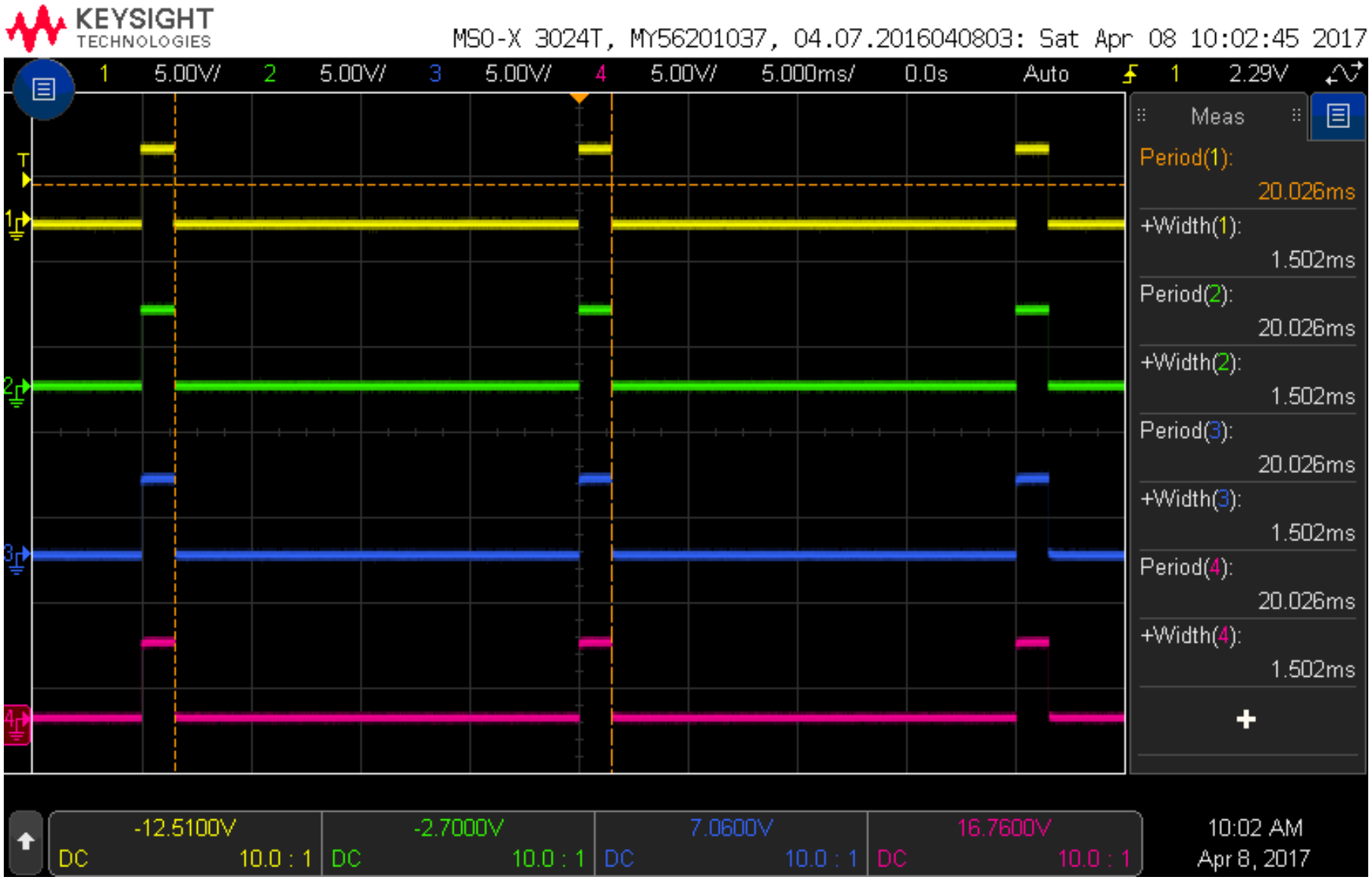
Example 9

4 servo motors (signal pins) are attached to PB0, PB1, PB2 and PB3 of ATmega328p. Assume that these servos require pulse width of 1.5ms to turn 90°. Use our servo class to make these servos turn 90°.

```
int main()
{
        Servo motor1('B', 0, 20000);
        Servo motor2('B', 1, 20000);
        Servo motor3('B', 2, 20000);
        Servo motor4('B', 3, 20000);

        motor1.Write(1500);
        motor2.Write(1500);
        motor3.Write(1500);
        motor4.Write(1500);

        while(1)
        {
            motor1.Refresh();
            motor2.Refresh();
            motor3.Refresh();
            motor4.Refresh();
        }
}
```

Driving 4 servos at the same time using the our own class (90° to all the servos)

The 4 control signals are in phase.



Example 10

A signal pin of a servo motor is attached to PB0 of ATmega328p. Write a program to make the servo rotate back and forth (sweeping) with the following parameters.

Minimum pulse width = 544us.
Maximum pulse width = 2400us.
Step size = 10us.
Delay = 25ms

Procedure:
Start at 544us pulse width.
Every 25ms, increase the pulse width by 10us until the max pulse width is reached.

After reaching the maximum pulse width,
Every 25ms, decrease the pulse width by 10us until the minimum pulse width is reached
Repeat the procedure.

```
int main()
{
	unsigned long last_time;
	int Min = 544;
	int Max = 2400;
	int Step = 10;
	int Delay_ms = 25;
	int Current = Min;

	Servo servo('B', 0, 20000);


	while(1)
	{
		unsigned long now = millis();
		if (now - last_time >= Delay_ms)
		{
			last_time = now;
			Current += Step;
			servo.Write(Current);
			if (Current >= Max || Current <= Min)
			{
			    Step = -Step;
			}
		}
		servo.Refresh();

	}
}
```

Important notes:

- Different servo motors have different min and max pulse widths. Read the datasheets.

- Do we assume *min pulse width* = 0° and *max pulse width* = 180°?

  No! It depends on the servo. Some servos are mirrored.

Lets make a servo sweeper class to help us easier to sweep multiple servos at the same time.

Servo sweeper class

A servo sweeper class can be constructed using the servo class.

```
class ServoSweeper
{
    private:
        Servo* servo;
        int Min, Max, Step, Delay_ms;
        int Current;
        unsigned long last_time=0;

    public:
        ServoSweeper()
        {

        }
        ServoSweeper(Servo* _servo, int _Min, int _Max, int _Step, int _Delay_ms)
        {
            Initialize(_servo, _Min, _Max, _Step, _Delay_ms);
        }
        void Initialize(Servo* _servo, int _Min, int _Max, int _Step, int _Delay_ms)
        {
            servo = _servo;
            Min = _Min;
            Max = _Max;
            Step = _Step;
            Delay_ms = _Delay_ms;
            Current = _Min;
        }
```

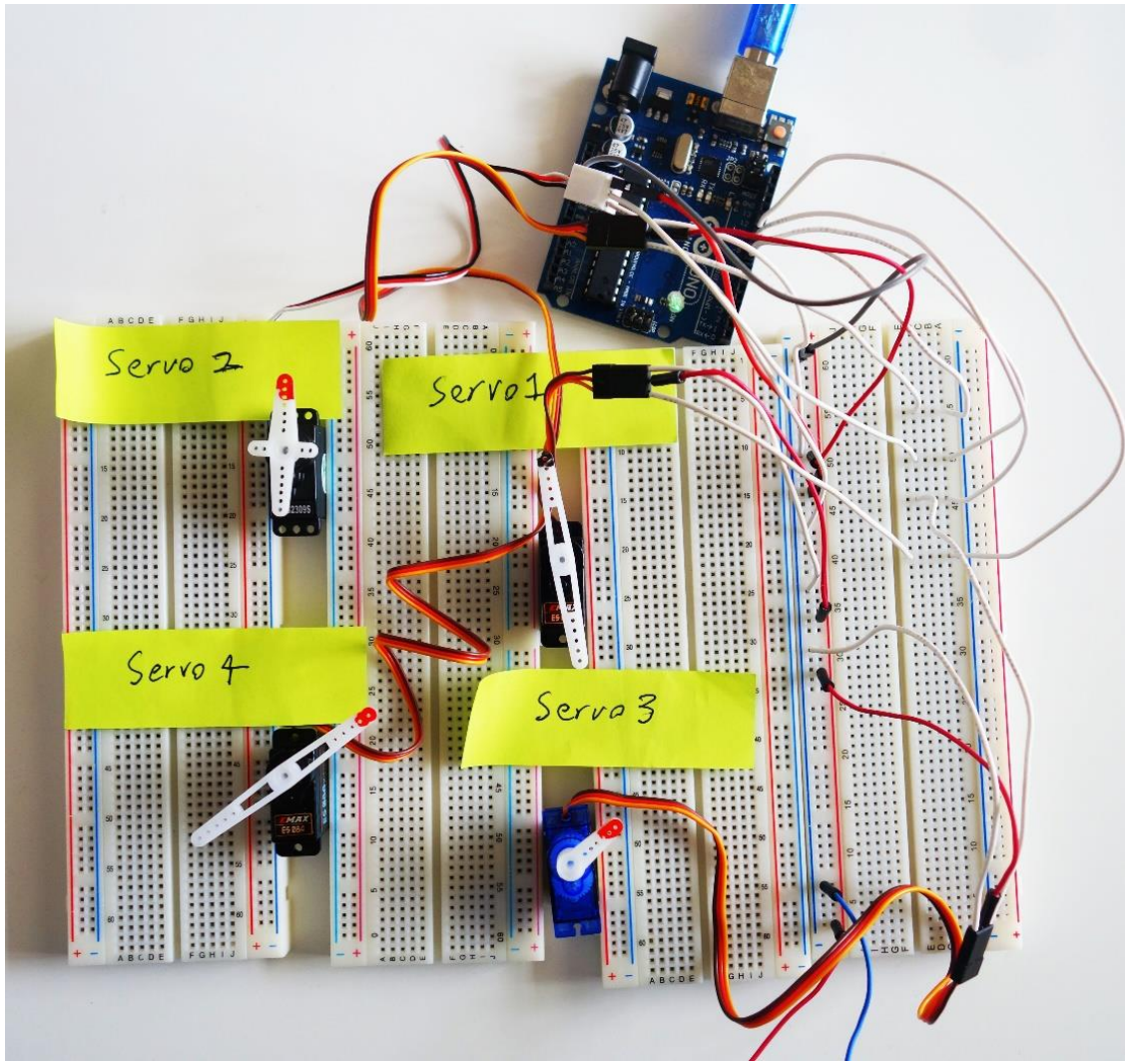
Continued next slide->

```
        void Refresh()
        {
              unsigned long now = millis();
              if (now - last_time >= Delay_ms)
              {
                    last_time = now;
                    Current += Step;
                    servo->Write(Current); //equivalent to (*servo).Write(Current);

                    if (Current >= Max || Current <= Min)
                    {
                        Step = -Step;
                    }
              }
              servo->Refresh();  //equivalent to (*servo).Refresh()
        }
};
```

Example 11

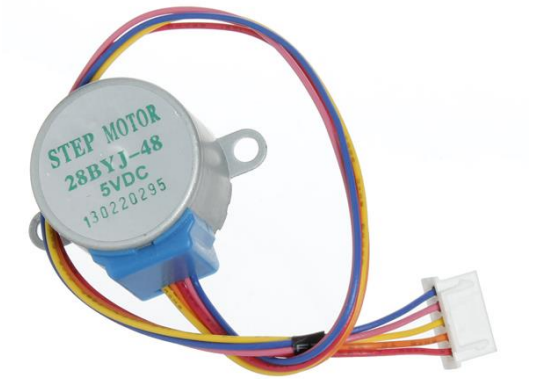
4 servo motors (Servo 1 to Servo 4) are connected to PB0 to PB3 of ATmega328p. Using Servo and ServoSweeper classes, write a program to make the 4 servo sweep according to the following specifications:

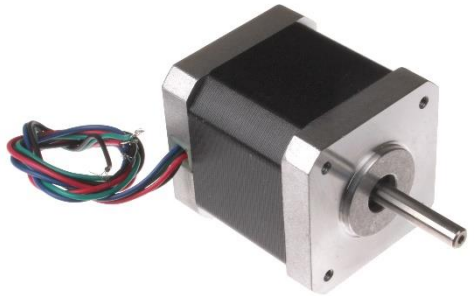
Servo	Period of control signal (µs)	Starting pulse width (µs)	Ending pulse width (µs)	Step (µs)	Delay (ms)
1	20,000	500	2,000	10	15
2	20,000	500	2,000	10	20
3	20,000	800	1,500	10	15
4	20,000	800	1,500	10	20



```
int main()
{
	Servo motor1('B', 0, 20000);
	Servo motor2('B', 1, 20000);
	Servo motor3('B', 2, 20000);
	Servo motor4('B', 3, 20000);

	ServoSweeper a(&servo1, 500, 2000, 10, 15);
	ServoSweeper b(&servo2, 500, 2000, 10, 20);
	ServoSweeper c(&servo3, 800, 1500, 10, 15);
	ServoSweeper d(&servo4, 800, 1500, 10, 20);

	while(1)
	{
		a.Refresh();
		b.Refresh();
		c.Refresh();
		d.Refresh();
	}
}
```

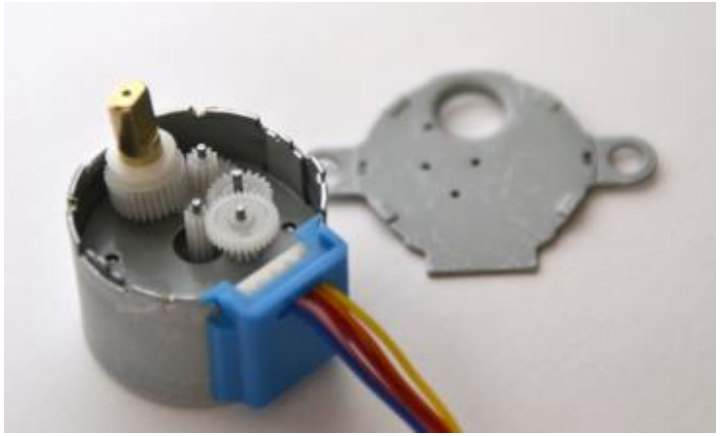
Stepper Motors

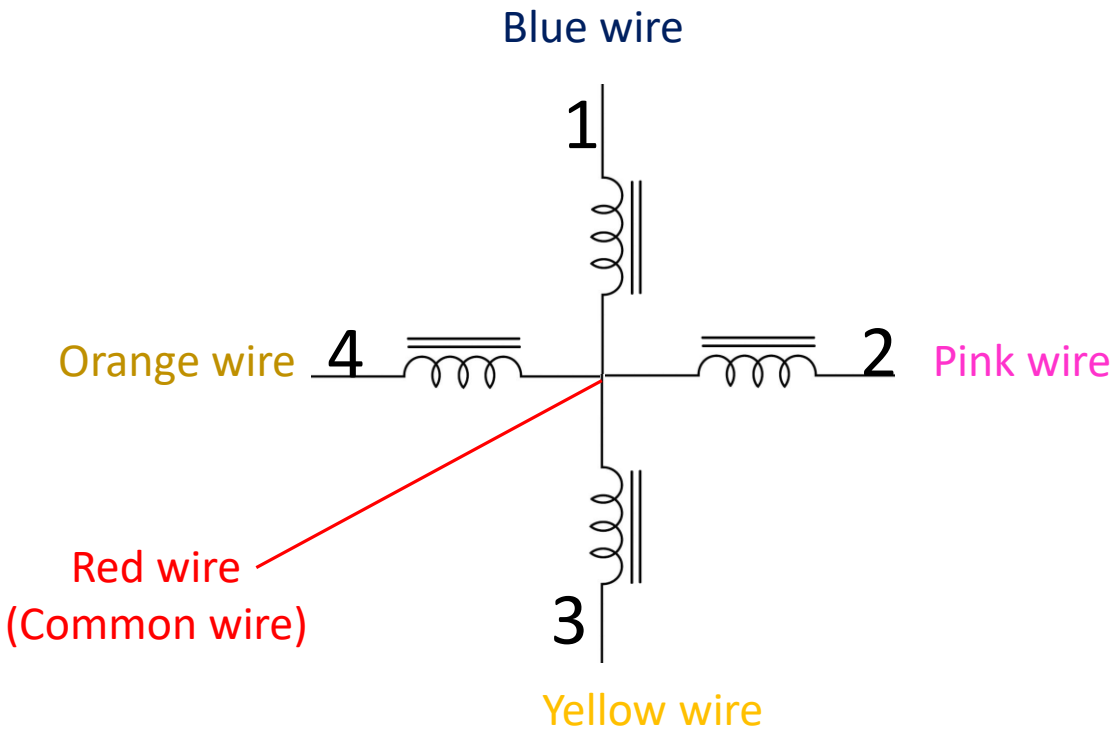




- There are many types of stepper motors with different number of wires.
- Bipolar stepper motors need H-bridges to reverse polarity.

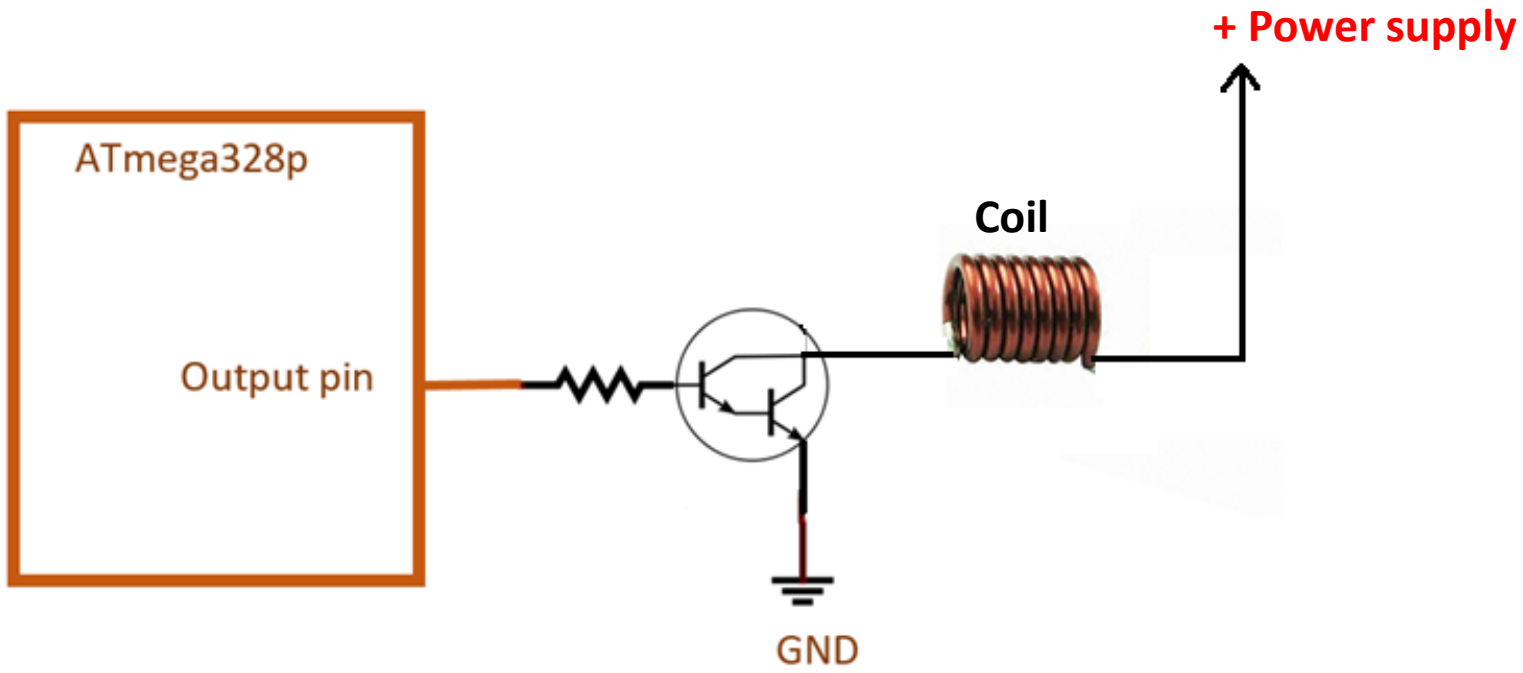
- **28BYJ-48** is a famous low-cost 5-wire unipolar stepper motor.



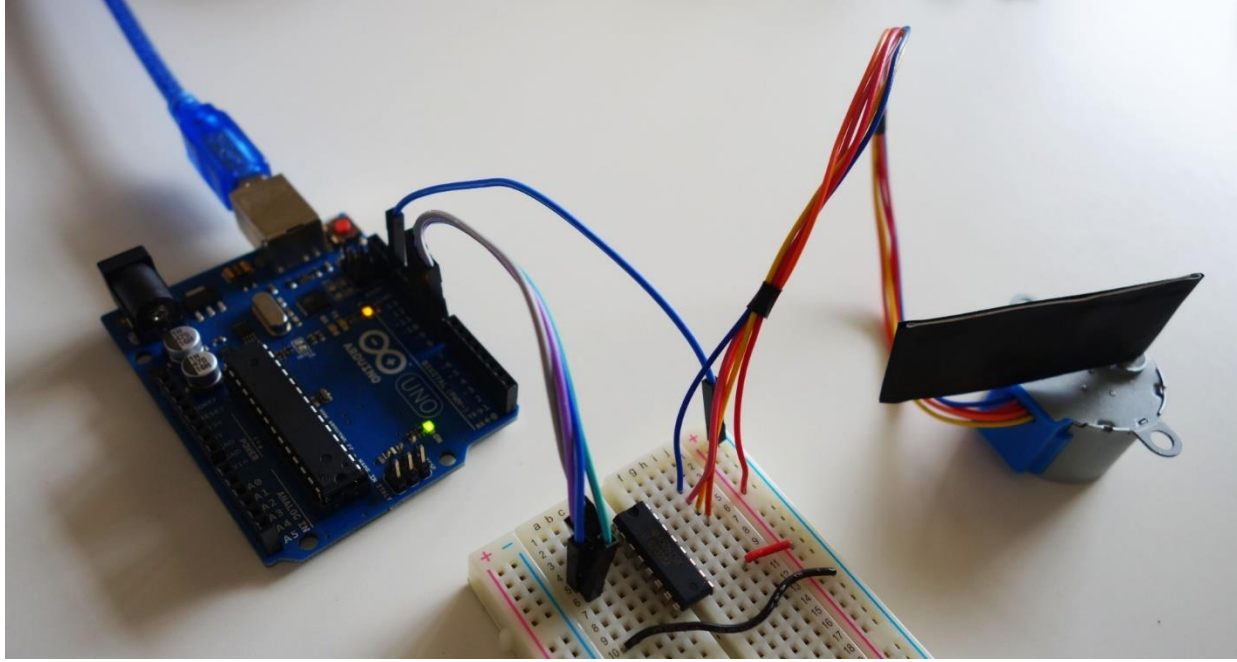


Darlington array

- Darlington array ICs (such as ULN2003) are commonly used with stepper motors.
- ULN2003 contains Darlingon transistors in an **open-collector** configuration
- When the output pin of the microcontroller is HIGH, one side of the coil gets shorted to GND.
- The other side of the coil needs to be connected to positive power supply.

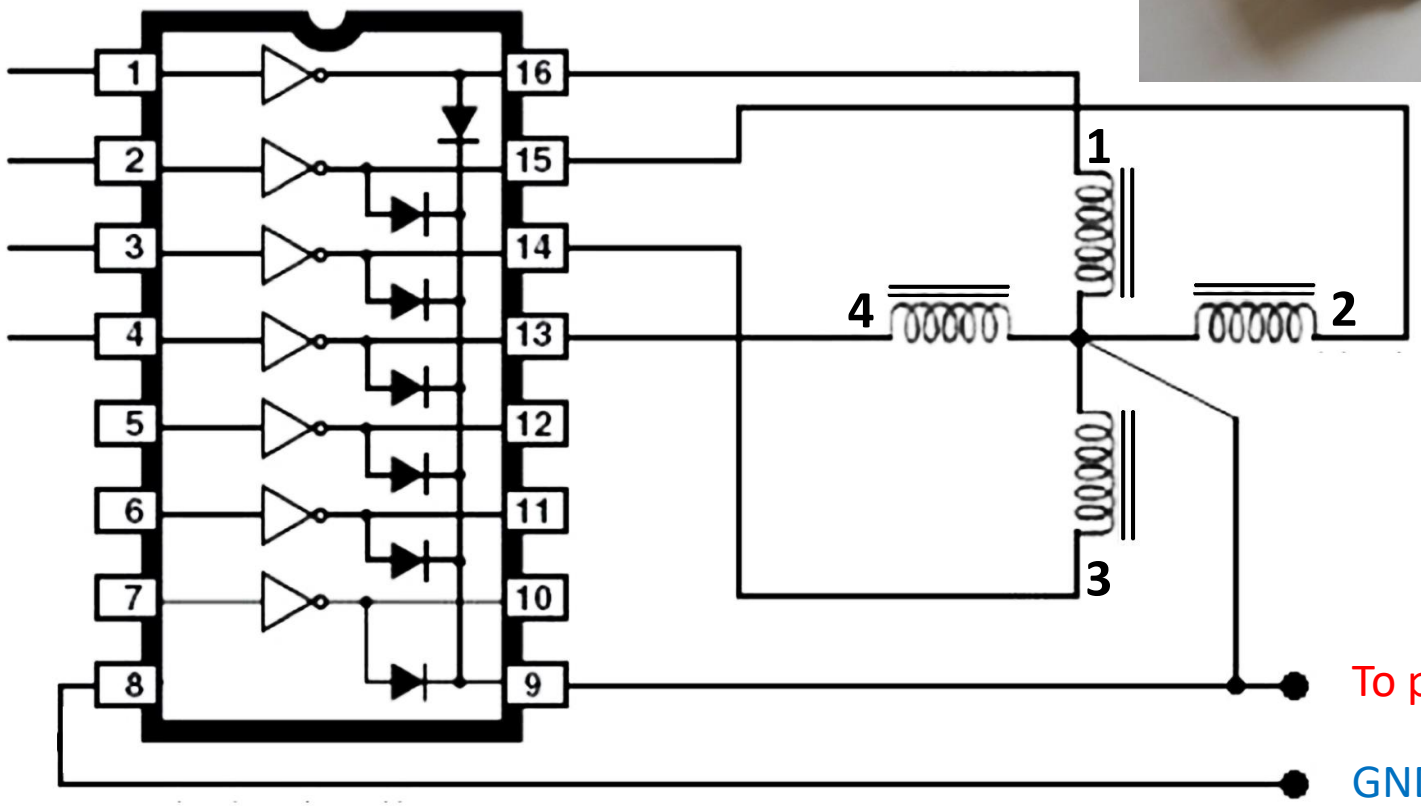


Schematic



ULN2003

To GPIO



To positive power supply

GND

Drive modes

- Wave drive
- Full drive
- Half-step drive
- Micro-stepping

Refer to:
https://www.youtube.com/watch?v=TWMai3oirnM&t=127s

Wave Drive

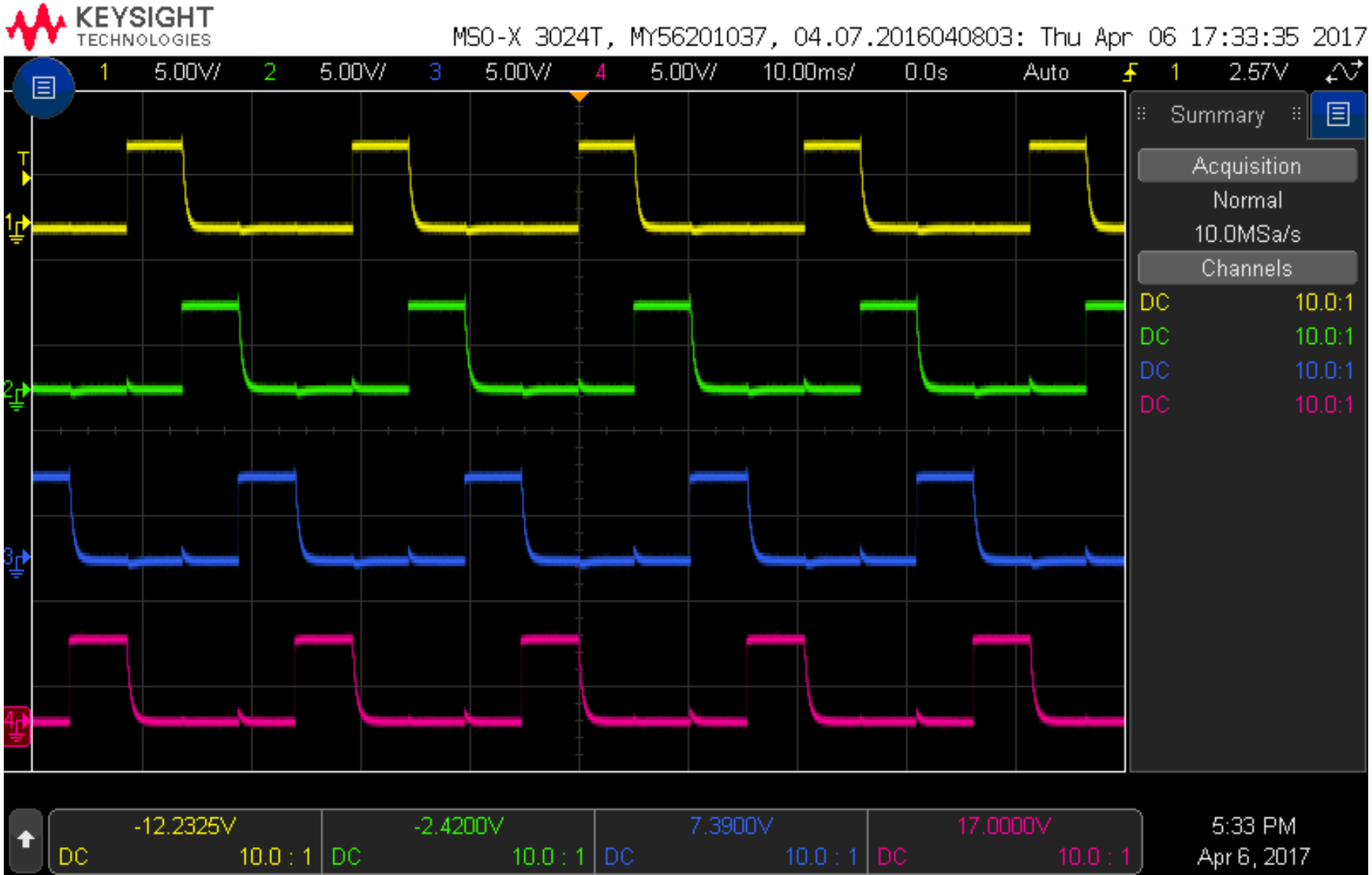
Coil voltage activation pattern

Coil 1

Coil 2

Coil 3

Coil 4



Full Step Drive

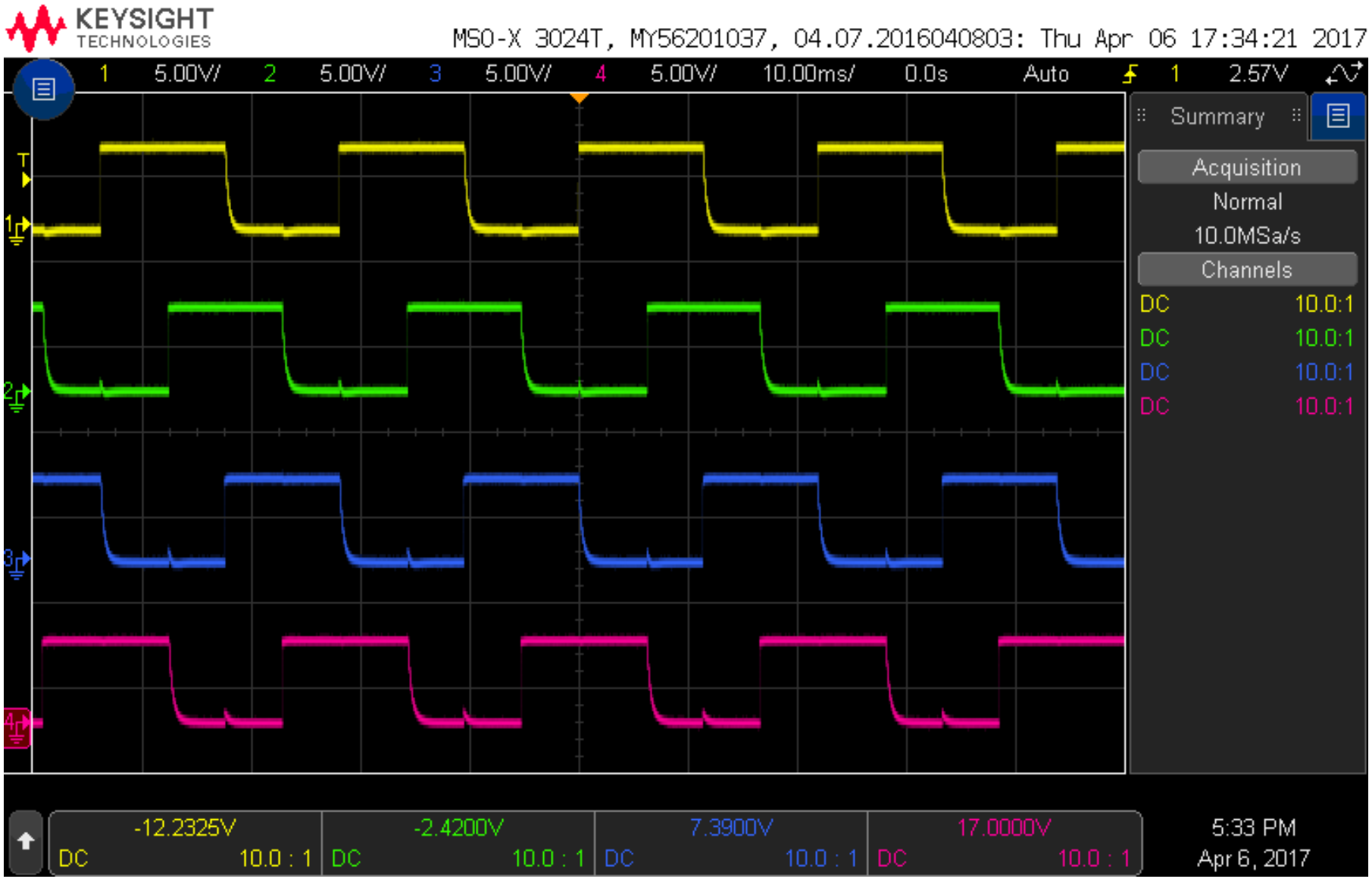
Coil voltage activation pattern

Coil 1

Coil 2

Coil 3

Coil 4



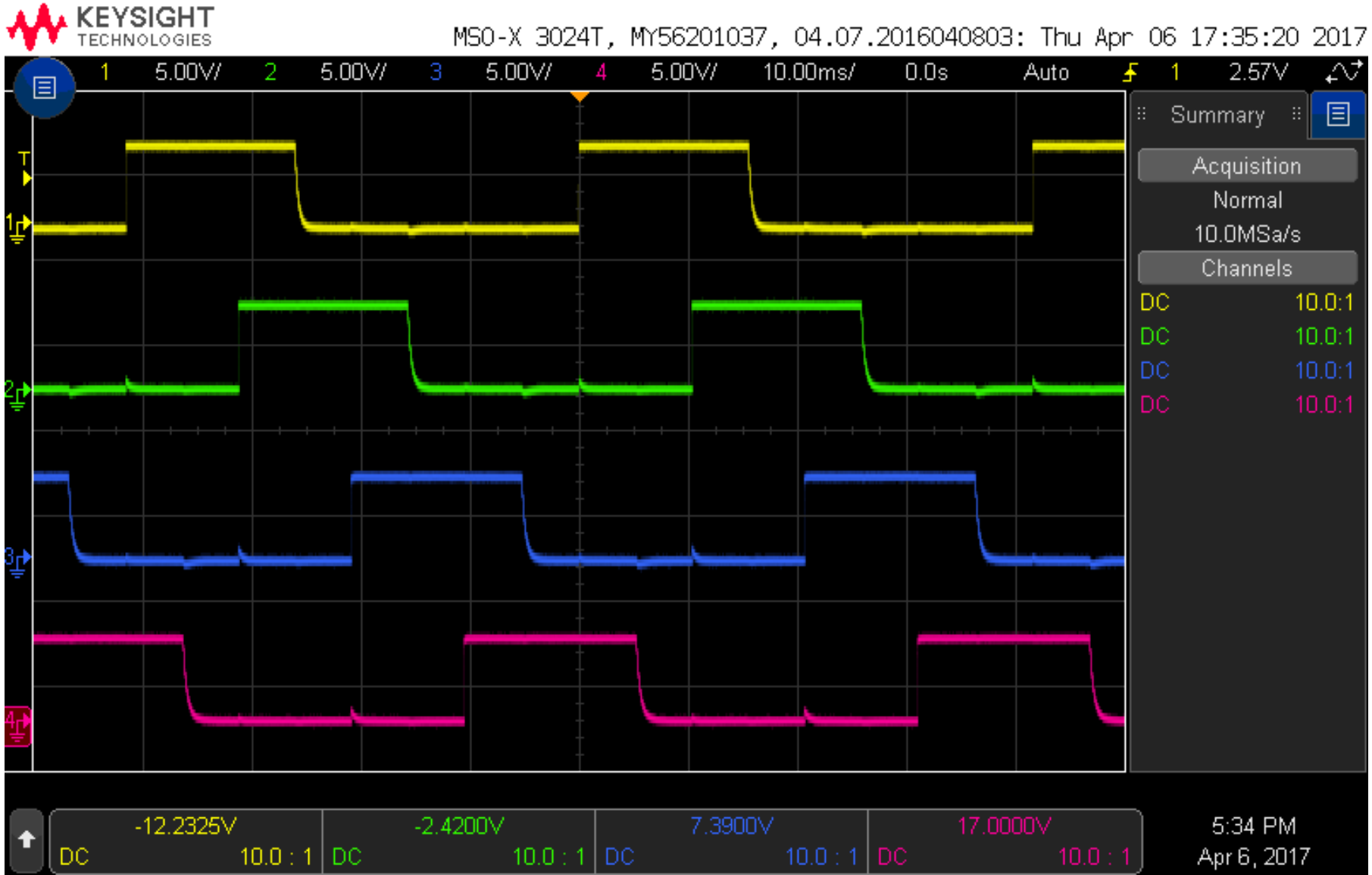
Half-step Drive *Coil voltage activation pattern*

Coil 1

Coil 2

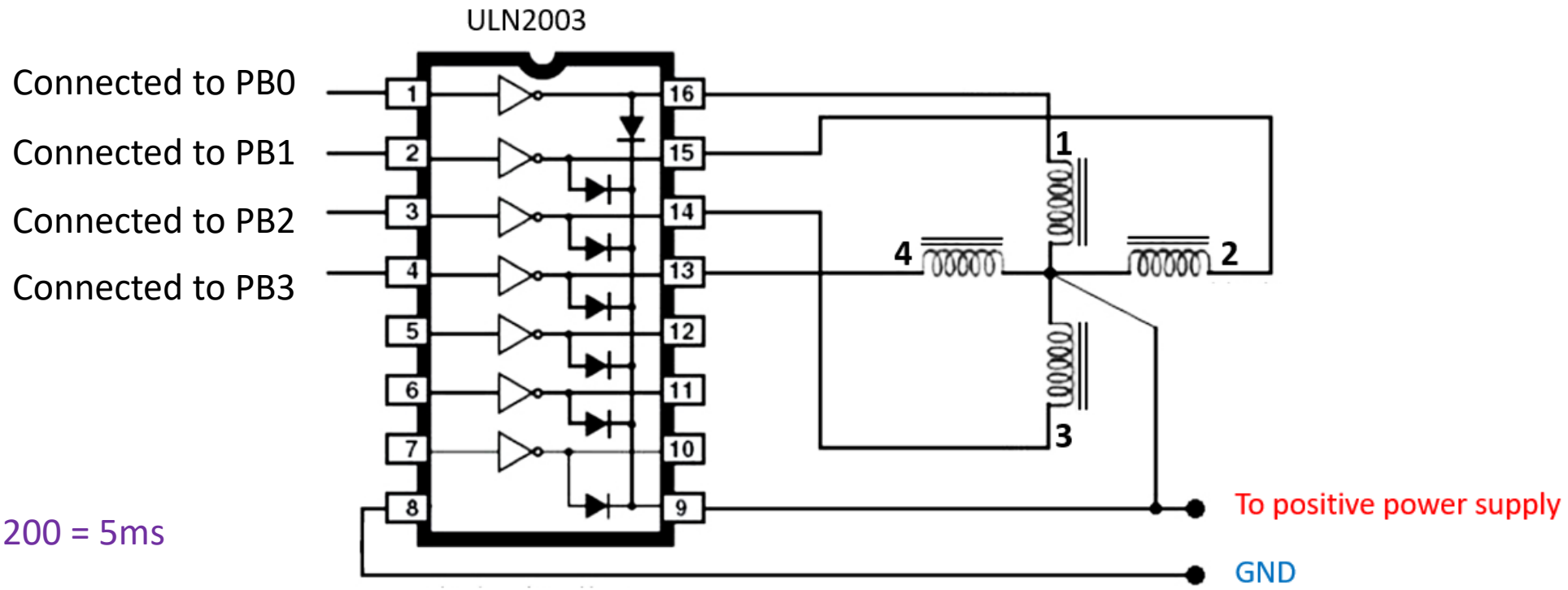
Coil 3

Coil 4



Example 12 (Wave drive - clockwise)

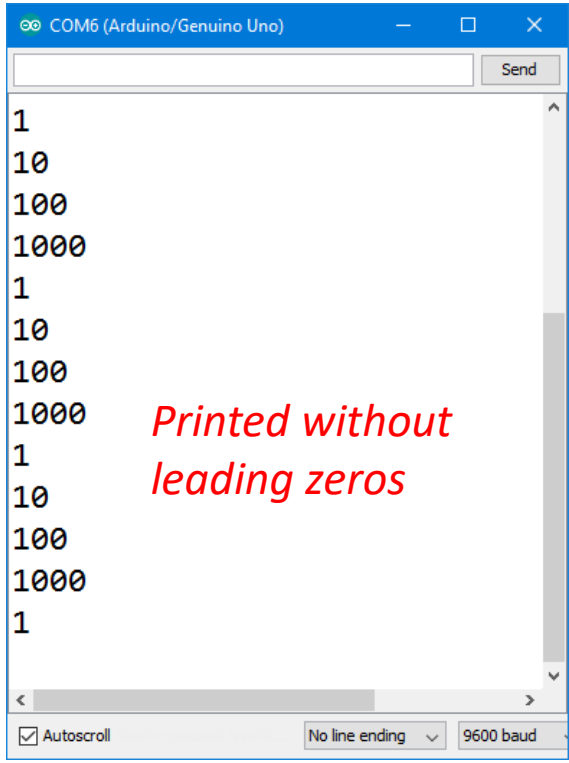
A 5-wire unipolar stepper motor is connected to ATmega328p via ULN2003 as shown in the following schematic. Write a program to make the stepper motor rotate in the clockwise direction at a rate of 200 steps/s. Use wave-drive mode.



Step delay = 1/ 200 = 5ms

Using the delay function

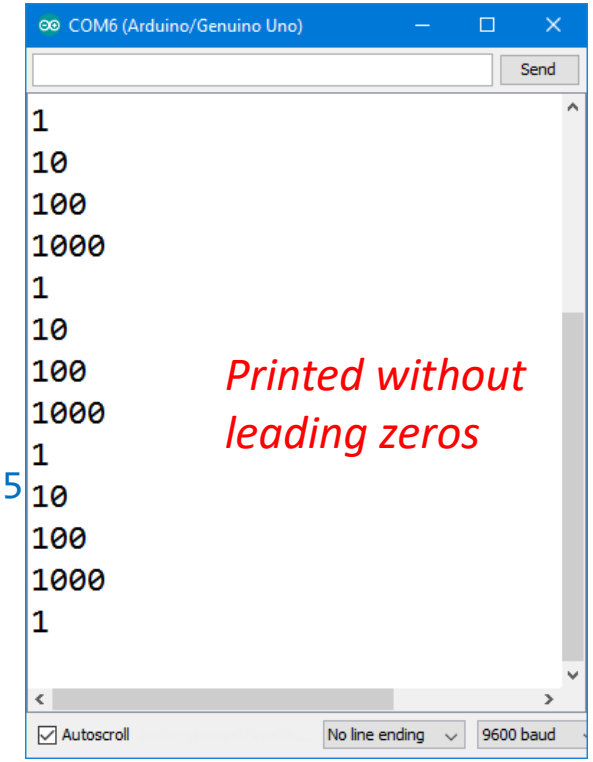
```
int main()
{
      char* ddrb = (char*) 0x24;                  //Points to DDRB register
      char* portb = (char*) 0x25;                 //Points to PORB register
      *ddrb = 0b00001111;                         //Set the 4 pins as output

      char DriveSteps[] = {1, 2, 4, 8};           //Activation pattern of wave drive
      char position=0;

      Serial.begin(9600);

      while(1)
      {

            *portb = DriveSteps[position];
            position++;
            if (position>=4)
            {
                  position = 0;
            }

            Serial.println((unsigned char)*portb, BIN);  //Serial printing is just for trouble shooting
            delay(5);                                     //Delay for 5ms before stepping again

      }
}
```



Printed without leading zeros

Without the delay function

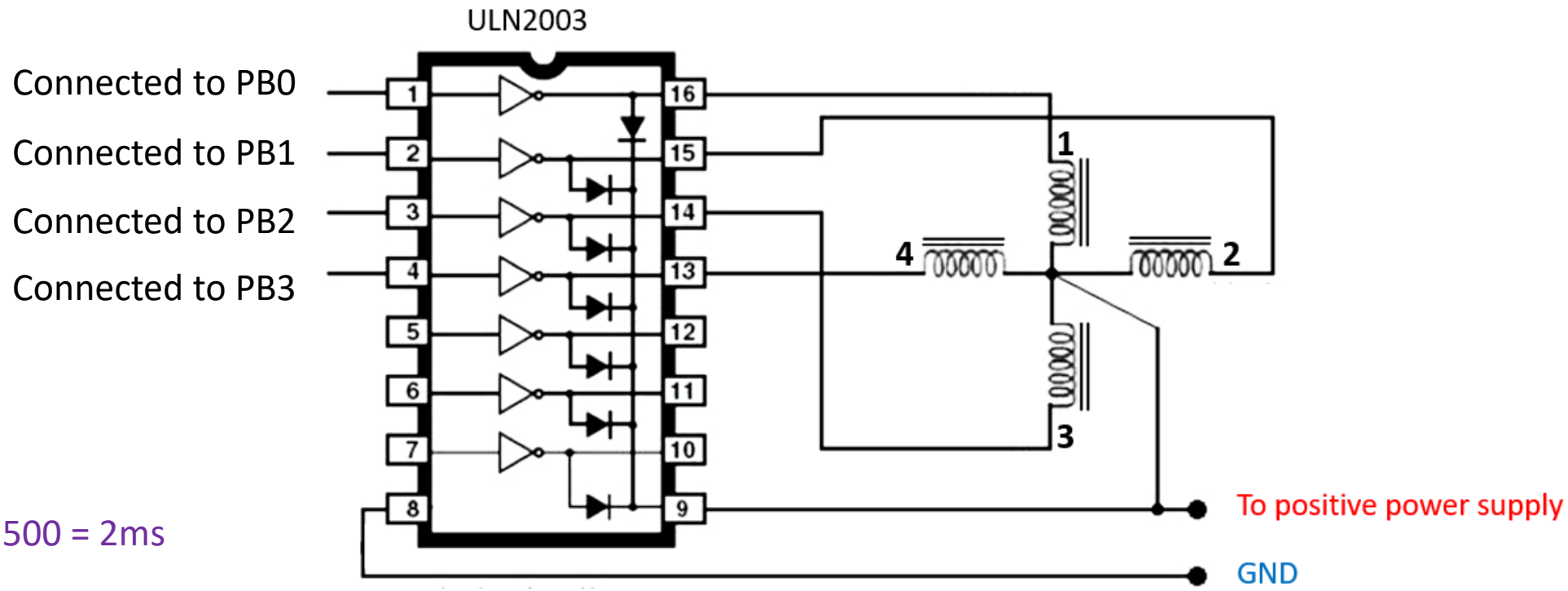
```
int main()
{
      char* ddrb = (char*) 0x24;              //Points to DDRB register
      char* portb = (char*) 0x25;            //Points to PORTB register
      *ddrb = 0b00001111;                    //Set the 4 pins as output

      char DriveSteps[] = {1, 2, 4, 8};     //Activation pattern of wave drive
      char position=0;

      Serial.begin(9600);
      unsigned long last;                    //Stores last time the motor moved

      while(1)
      {
            unsigned long now = millis();
            if (now-last>=5)                          //If the motor has not moved for 5
            {

                  *portb = DriveSteps[position];

                  position++;
                  if (position>=4)
                  {
                      position = 0;
                  }
                  last = now;
                  Serial.println((unsigned char)*portb, BIN); //Serial printing is just for trouble shooting

             }
      }

}
```



Printed without leading zeros

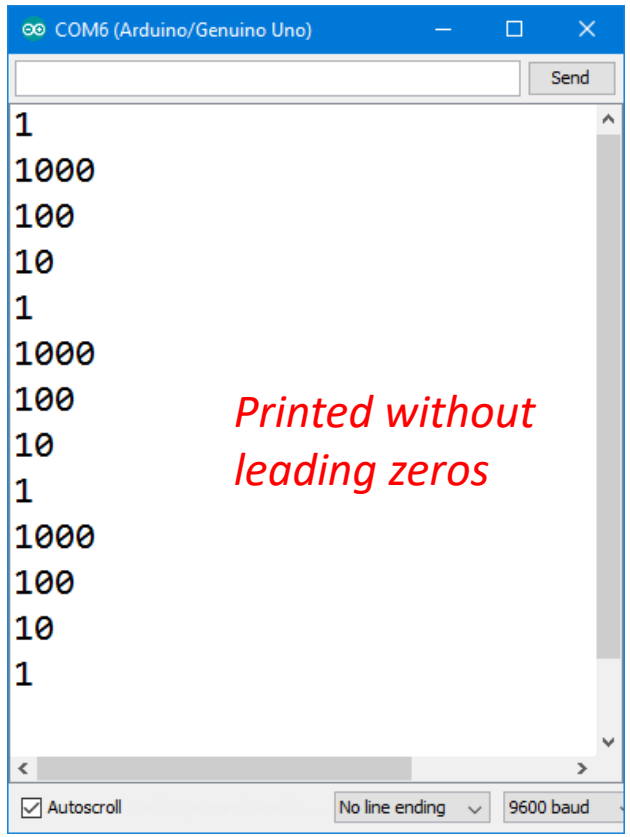
Example 13 (Wave drive - counterclockwise)

Same as the previous example. But this time, make the stepper motor rotate in the counterclockwise direction at a rate of 500 steps/s.



Step delay = 1/ 500 = 2ms

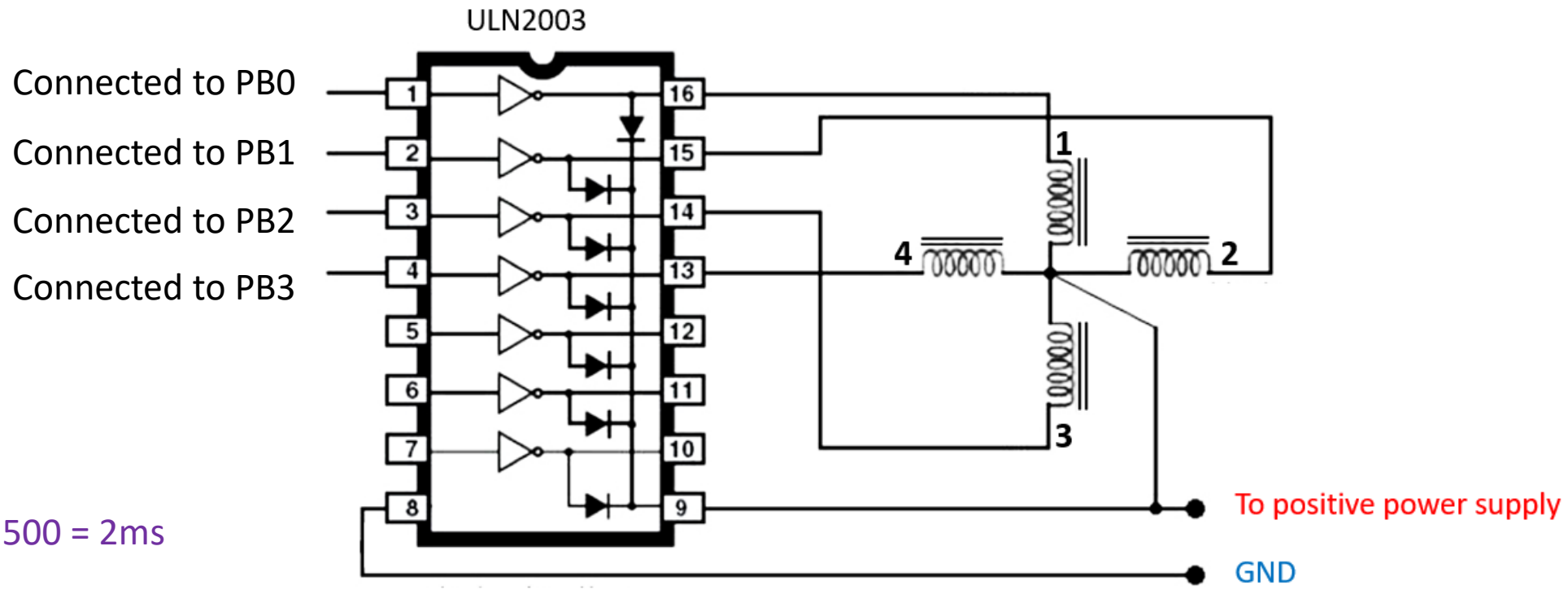
```
int main()
{
      char* ddrb = (char*) 0x24;              //Points to DDRB register
      char* portb = (char*) 0x25;             //Points to PORTB register
      *ddrb = 0b00001111;                     //Set the 4 pins as output

      char DriveSteps[] = {1, 2, 4, 8};        //Activation pattern of wave drive
      char position=0;

      Serial.begin(9600);
      unsigned long last;                      //Stores last time the motor moved

      while(1)
      {
            unsigned long now = millis();
            if (now-last>=2)                    //If the motor has not moved for 2 ms
            {

                  *portb = DriveSteps[position];

                  position--;
                  if (position<0)
                  {
                      position = 3;
                  }
                  last = now;
                  Serial.println((unsigned char)*portb, BIN); //Serial printing is just for trouble shooting

             }
      }

}
```



Printed without leading zeros

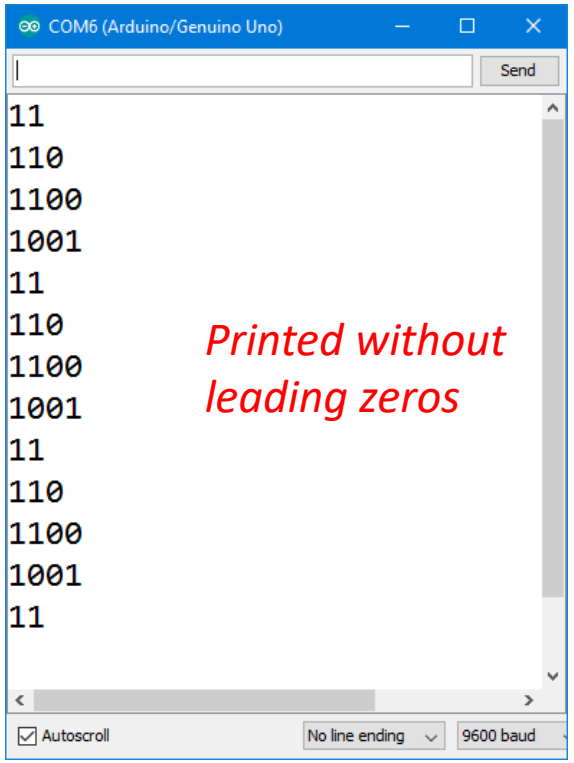
Example 14 (Full drive)

Same setup as the previous examples. But this time, use full drive mode. Make make the stepper motor rotate in the clockwise direction at a rate of 500 steps/s.



Step delay = 1/ 500 = 2ms

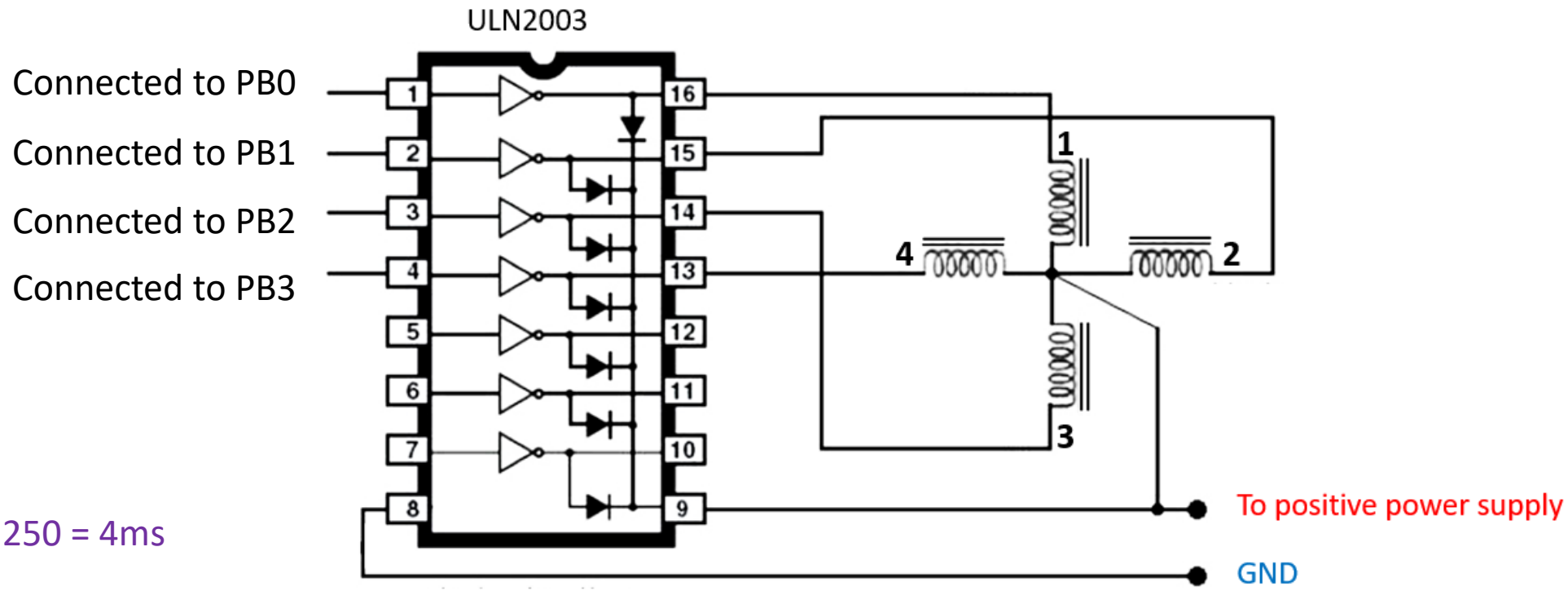
```
int main()
{
      char* ddrb = (char*) 0x24;              //Points to DDRB register
      char* portb = (char*) 0x25;             //Points to PORTB register
      *ddrb = 0b00001111;                     //Set the 4 pins as output

      char DriveSteps[] = {3, 6, 12, 9};       //Activation pattern of full drive
      char position=0;

      Serial.begin(9600);
      unsigned long last;                      //Stores last time the motor moved

      while(1)
      {
            unsigned long now = millis();
            if (now-last>=2)                    //If the motor has not moved for 2 ms
            {

                  *portb = DriveSteps[position];

                  position++;
                  if (position>=4)
                  {
                      position = 0;
                  }
                  last = now;
                  Serial.println((unsigned char)*portb, BIN); //Serial printing is just for trouble shooting

             }
      }

}
```



Printed without leading zeros

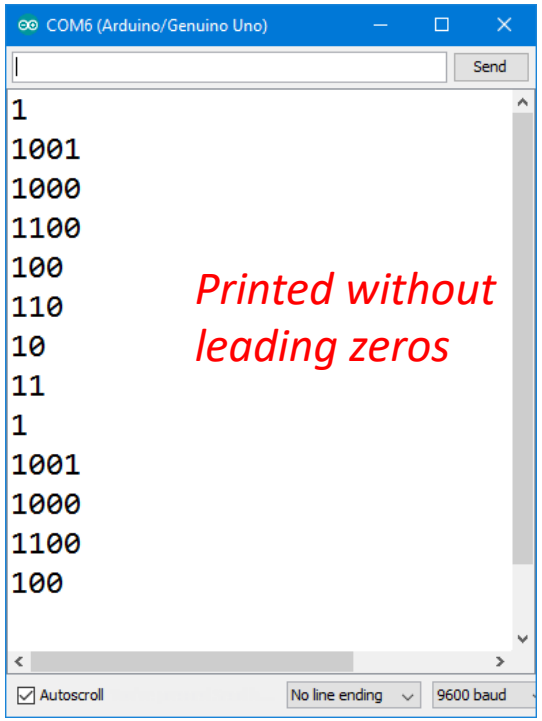
Example 15 (Half-step drive)

Same setup as the previous examples. But this time, use half-step drive mode. Make make the stepper motor rotate in the counterclockwise direction at a rate of 250 steps/s.



Step delay = 1/ 250 = 4ms

```
int main()
{
      char* ddrb = (char*) 0x24;              //Points to DDRB register
      char* portb = (char*) 0x25;             //Points to PORTB register
      *ddrb = 0b00001111;                     //Set the 4 pins as output

      char DriveSteps[] = {1, 3, 2, 6, 4, 12, 8, 9}; //Activation pattern of half-step drive
      char position=0;

      Serial.begin(9600);
      unsigned long last;                      //Stores last time the motor moved

      while(1)
      {
            unsigned long now = millis();
            if (now-last>=4)                    //If the motor has not moved for 2 ms
            {

                  *portb = DriveSteps[position];

                  position--;
                  if (position<0)
                  {
                      position = 7;
                  }
                  last = now;
                  Serial.println((unsigned char)*portb, BIN); //Serial printing is just for trouble shooting

             }
      }

}
```



Printed without leading zeros

A stepper class that handles different drive modes and ports can be created.

```
class Stepper
{
    private:
        char* DDR;
        char* PORT;
        char DriveSteps[8];
        char Position;
        char MaximumPosition;

    public:
        Stepper(char port, char mode)
        {
              Position = 0;

              switch (port)
              {
                  case 'B':
                    DDR = (char*) 0x24;
                    PORT = (char*) 0x25;
                    break;
                  case 'C':
                    DDR = (char*) 0x27;
                    PORT = (char*) 0x28;
                    break;
                  case 'D':
                    DDR = (char*) 0x2A;
                    PORT = (char*) 0x2B;
                    break;
              }
              *DDR |= 0b00001111;
```

continued

```
              switch (mode)
              {
                  case 'W':
                      MaximumPosition = 4;
                      DriveSteps[0] = 1;
                      DriveSteps[1] = 2;
                      DriveSteps[2] = 4;
                      DriveSteps[3] = 8;
                      break;
                  case 'F':
                      MaximumPosition = 4;
                      DriveSteps[0] = 3;
                      DriveSteps[1] = 6;
                      DriveSteps[2] = 12;
                      DriveSteps[3] = 9;
                      break;
                  case 'H':
                      MaximumPosition = 8;
                      DriveSteps[0] = 1;
                      DriveSteps[1] = 3;
                      DriveSteps[2] = 2;
                      DriveSteps[3] = 6;
                      DriveSteps[4] = 4;
                      DriveSteps[5] = 12;
                      DriveSteps[6] = 8;
                      DriveSteps[7] = 9;
                      break;
              }
        }
```

```
        void DriveCW()
        {
                *PORT = DriveSteps[Position];
                Position++;
                if (Position>=MaximumPosition)
                {
                        Position = 0;
                }
        }
        void DriveCCW()
        {
                *PORT = DriveSteps[Position];
                Position--;
                if (Position<0)
                {
                        Position = MaximumPosition-1;
                }
        }
};
```

```
class StepperRotator
{
      private:
          Stepper* stepper;
          unsigned long delay_us;
          bool clockwise;
          unsigned long previous=0;

      public:
          StepperRotator(Stepper* _stepper, bool _clockwise, unsigned long _delay_us)
          {
              stepper = _stepper;
              delay_us = _delay_us;
              clockwise = _clockwise;
          }
          void Refresh()
          {
              unsigned long now = micros();
              if (now-previous>=delay_us)
              {
                  if (clockwise)
                  {
                      stepper->DriveCW();
                  }
                  else
                  {
                      stepper->DriveCCW();
                  }
                  previous = now;
              }
          }

};
```

A class for rotating stepper motors can also be created.

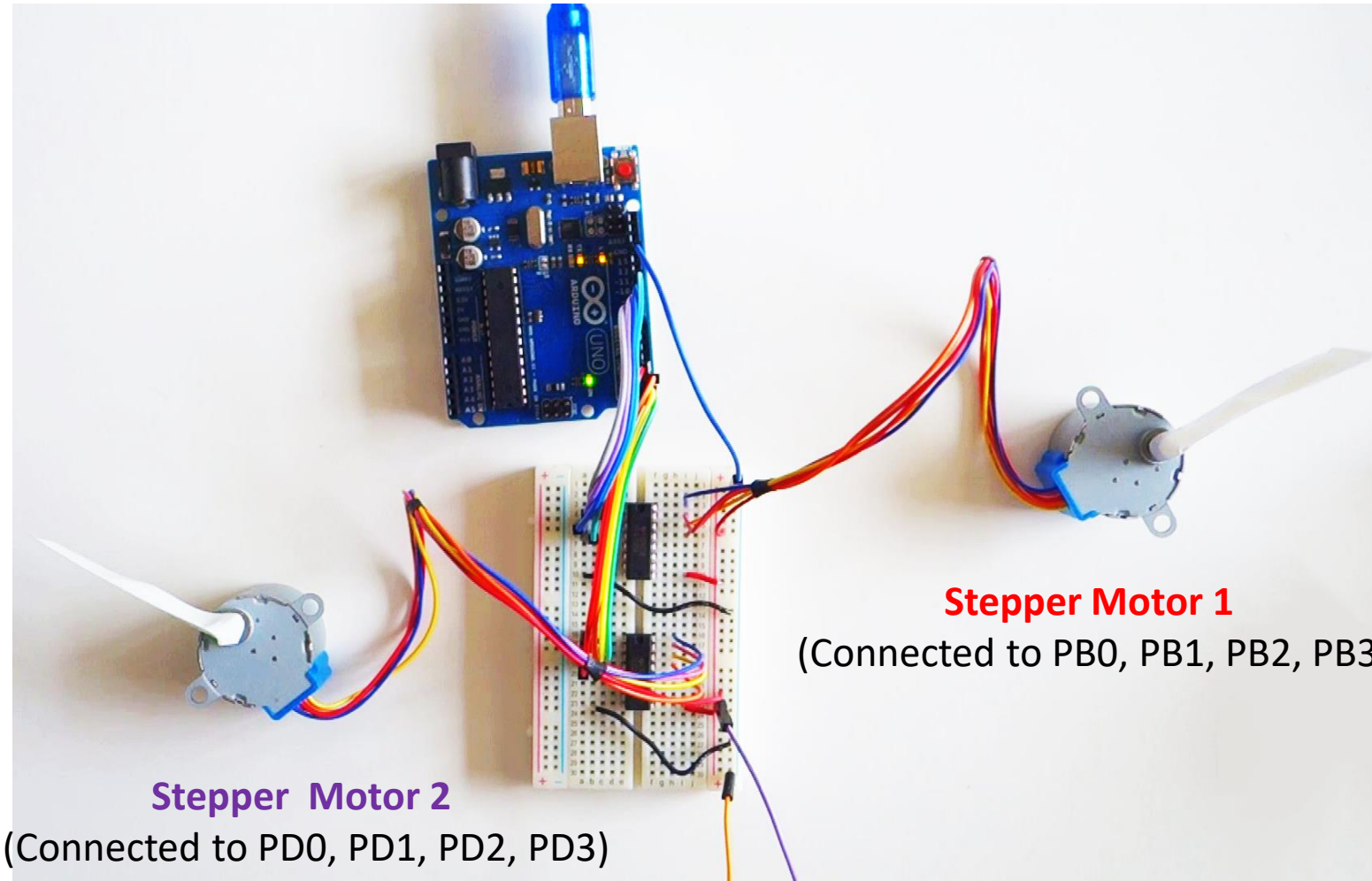
Example 16

Two unipolar stepper motors are connected to PORTB (0 to 3) and PORTD (0 to 3) of ATmega328p.

Stepper motor 1 needs to be rotating clockwise at a rate of 500 steps/s using full-step drive mode

Stepper motor 2 needs to be rotating counterclockwise at a rate of 500 steps/s using half-step drive mode

Program the microcontroller to achieve this.



```
int main()
{
	Stepper stepper1('B', 'F');
	Stepper stepper2('D', 'H');

	StepperRotator rotator1(&stepper1, 1, 2000);
	StepperRotator rotator2(&stepper2, 0, 2000);

	while(1)
	{
		rotator1.Refresh();
		rotator2.Refresh();
	}
}
```

Controlling many motors simultaneously

Performance can degrade if many motors have to be controlled simultaneously.

Solutions:

1) Use hardware-based motor drivers
2) Instead of a microcontroller, use a single-board computer (with RTOS if possible) or an FPGA
3) Use decentralized architecture

Example 1:

Spider robot (18 servo motors)
https://www.youtube.com/watch?v=Fe1PCxCGbqs

Example 2:

Atlas Robot by Boston Dynamics
https://www.youtube.com/watch?v=rVlhMGQgDkY

Decentralized Architecture

In a decentralized architecture, low-level motor control tasks are delegated to slaves. The Master commands the slaves through a communication bus.