ДСТУ 3008:2015

03022, м. Київ, пр. Академіка Глушкова 2, 044-526-4567

ЗВІТ

ПРО КОНКУРСНУ РОБОТУ

Отримання максимальної можливої частоти на платі Arduino Uno

(підпис)

(дата)

2020

**СПИСОК АВТОРІВ**

Студент В. О. Кришталь

(підпис)

(дата)

**РЕФЕРАТ**

Звіт про НДР: 23 ст., 1 ч., 21 рис.

Об’єкт дослідження – плата Arduino Uno Rev3

Мета робота – отримати сигнал максимальної частоти за допомогою плати Arduino з тактовою частотою 16 МГц

Методи дослідження – симуляція мікроконтролера в програмі Proteus Design Suite, дослідження сигналів на осцилографі

**Зміст**

[1 Вступ та теоретичні відомості 5](#_Toc34689059)

[2 Отримання максимальної частоти методом простого повторення 6](#_Toc34689060)

[2.1 Перша спроба – використання digitalWrite 6](#_Toc34689061)

[2.2 Друга спроба – використання регістрів 8](#_Toc34689062)

[2.3 Остаточна реалізація 9](#_Toc34689063)

[2.4 Висновки 12](#_Toc34689064)

[3 Другий метод – використання вбудованого таймеру та PWM 12](#_Toc34689065)

[3.1 Принцип роботи таймеру та PWM 12](#_Toc34689066)

[3.2 Реалізація 13](#_Toc34689067)

[3.3 Висновки 17](#_Toc34689068)

[4 Зміна fuse бітів 17](#_Toc34689069)

# 1 Вступ та теоретичні відомості

У даній роботі розглянуто задачу отримання сигналу максимальної частоти за допомогою Arduino Uno Rev3. На цій платі встановлено мікроконтролер ATMega328P [1]. За допомогою зовнішнього кварцового осцилятора досягається тактова частота МГц. Документацію на мікроконтролер можна знайти за посиланням [2]. У роботі використовувалася програма Proteus Design Suite для моделювання мікроконтролера та дослідження сигналів.

Очевидно, що частота сигналу, отриманого на виході мікроконтролера, не може перевищувати його тактову частоту, що ставить теоретичний ліміт Його можна досягти, вивівши тактовий сигнал напряму ([2] 8.9), але це потребує використання зовнішнього програматора, щоб модифікувати так звані fuse біти ([2] 27.2). Максимальна ж частота, яку можна досягти без додаткових пристроїв, використовуючи лише плату Arduino, становить . У роботі розглянуто та порівняно два методи її досягнення. Перший полягає в тому, щоб програмно виставляти вихідний сигнал на одній з ніжок почергово то на логічну одиницю (5 В), то на логічний нуль (0 В). На кожну з цих дій іде одна процесорна інструкція, тому повний період займає дві інструкції, і частота рівна половині тактової. Цей спосіб названо методом простого повторення. Другий спосіб використовує вбудований таймер і не може видати більшу частоту, бо це в ньому не передбачено. Він кращий за попередній, тому що при цьому процесор залишається вільним і може виконувати інші дії.

# 2 Отримання максимальної частоти методом простого повторення

## 2.1 Перша спроба – використання digitalWrite

Найочевиднішим способом отримання максимальної вихідної частоти є чередування команд digitalWrite(pin, HIGH) та digitalRead(pin, LOW), які є стандартними функціями для виводу цифрових сигналів в Arduino IDE. Для виводу використовується вихід 8 Arduino, що відповідає ніжці PB0 мікроконтролера [3]. Код програми зображено на рис. 2.1.

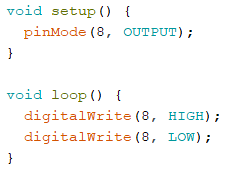
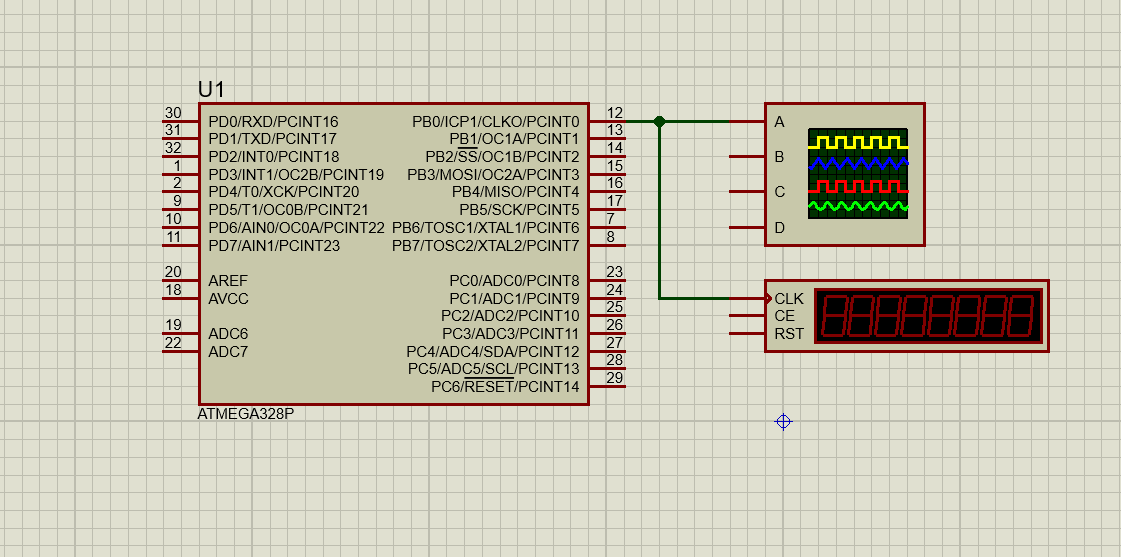


Рис. 2.1

Для моделювання в Proteus зібрана схема, зображена на рис 2.2. Для достатньо швидкої роботи симулятора тактова частота мікроконтролера виставлена рівною МГц. Для вимірювання частоти сигналу використовуються осцилограф та частотомір. Як видно з рис 2.3 вихідна частота складає приблизно .2 кГц, 109. Таке велике значення відношення пояснюється тим, що таблиця відповідності між виходами Arduino та ніжками мікроконтролера збережена в пам’яті самого чипа і функція digitalWrite при кожному виклику витрачає час на те, щоб дістати ці дані з пам’яті.

 Рис. 2.2.

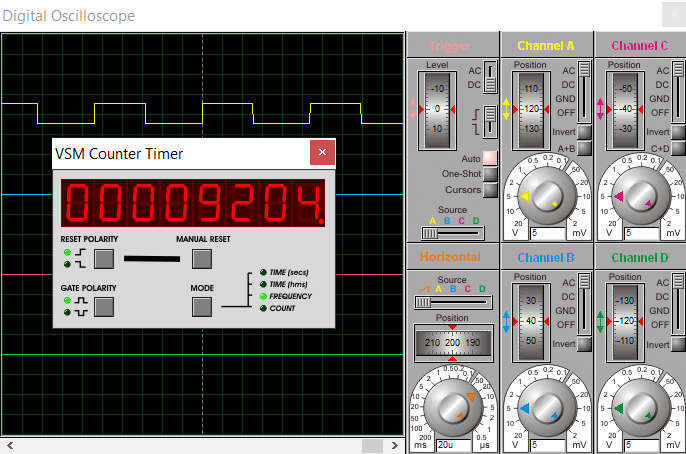


Рис. 2.3

Також вона перевіряє чи не використовується даний порт для PWM (pulse-width modulation – широтно-імпульсна модуляція). Все це можна побачити в її коді [4]. Таким чином можна дійти висновку, що вбудовані функції Arduino для виводу цифрових сигналів не є достатньо швидкими і потрібно використовувати інший метод.

## 2.2 Друга спроба – використання регістрів

Згідно з розділом 13 [2], виходами мікроконтролера можна керувати, змінюючи значення в регістрах PORTX, де X – літера, що відповідає порту, де знаходиться потрібна ніжка. Якщо записати у відповідний біт 1, то на виході з’явиться логічна одиниця, якщо ж записати туди 0 – логічний нуль.

Код відповідної програми зображений на рис. 2.4. У циклі в регістр PORTB

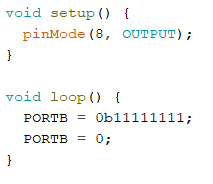


Рис. 2.3

записуємо спочатку байт, що складається з одних одиниць, потім нуль.

Результат симуляції зображений на рис. 2.4. кГц, 6. Як бачимо, логічна одиниця встановлюється всього на 1 мкс, тобто на один такт процесора, проте весь цикл займає 6 тактів. Причиною цього є те, що для виконання циклу, програма в кінці кожної його ітерації повинна повертатися

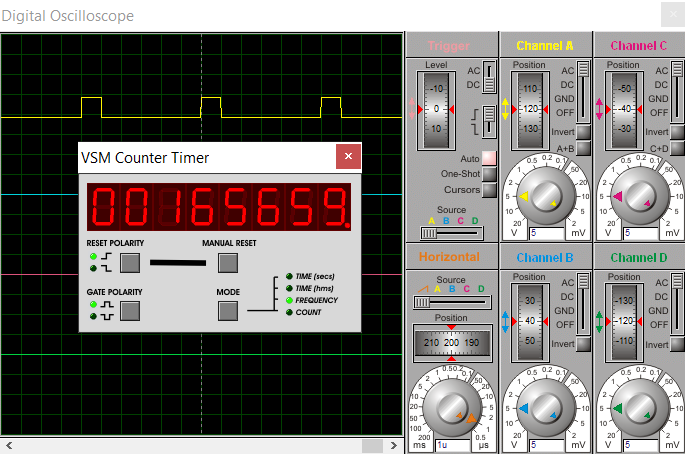


Рис. 2.4

в початок. Це потребує окремої процесорної інструкції, що займає час на виконання. Цю проблему можна вирішити, якщо повторювати команди ввімкнення та вимкнення порту багато разів всередині циклу. Тоді повернення до початку циклу буде відбуватись лише раз на декілька періодів сигналу і тому буде мати набагато менший ефект.

## 2.3 Остаточна реалізація

Для того, щоб мінімізувати час, затрачений на перехід до початку циклу, команди для ввімкнення та вимкнення портів були повторені 100 разів. Щоб не писати код вручну, були створені макроси repeat\_ten та repeat\_hundred, які повторюють шматок коду відповідно 10 та 100 разів (рис 2.5). Результат симуляції показаний на рис. 2.6.

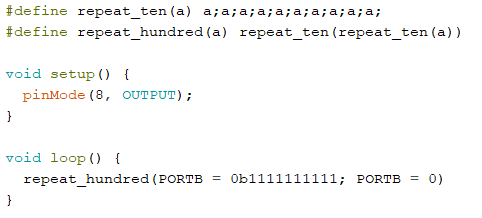


Рис. 2.5

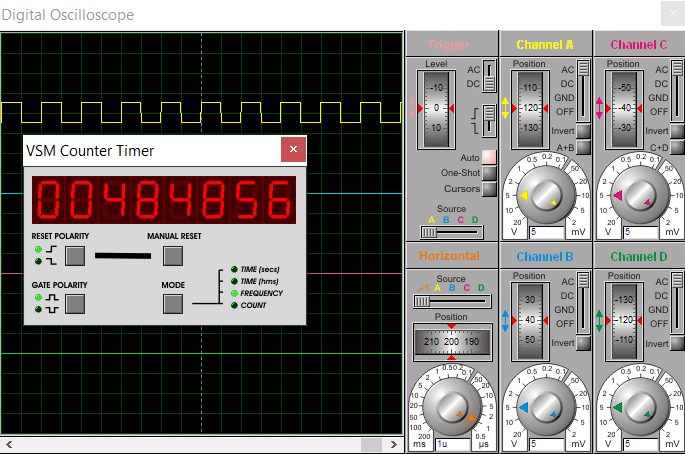


Рис. 2.6

Як бачимо, кГц, , тобто період сигналу рівний двом тактам процесору і таким чином був досягнутий теоретичний максимум. Частоту сигналу можна легко регулювати, повторюючи команди різне число разів, і навіть робити тривалість логічних одиниці та нуля в сигналі різними, але при цьому частота завжди буде кратна тактовій. Код модифікованого варіанту цієї програми, що демонструє цю можливість, зображений на рис. 2.7.

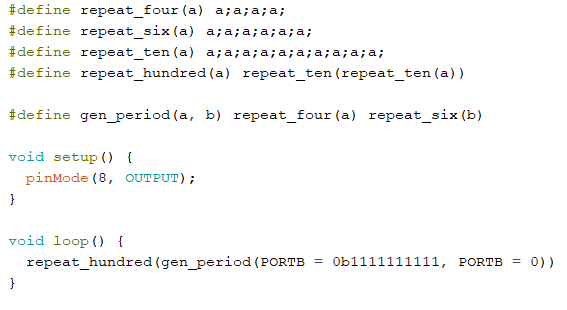


Рис. 2.7

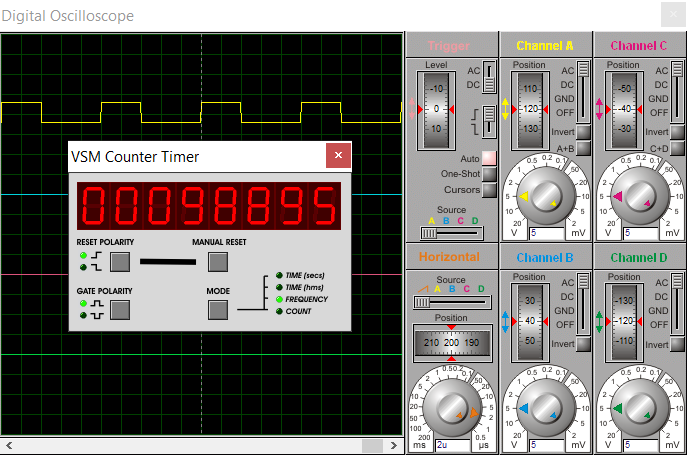


Рис. 2.8

Він створює сигнал з , де періоду відображається логічна одиниця, а періоду – логічний нуль.

## 2.4 Висновки

З отриманих результатів можна зробити висновок, що за допомогою методу простого повтору можна отримати максимальну теоретично можливу частоту вихідного сигналу, а також будь-яку кратну їй частоту. Перевагою цього методу є те, що можна регулювати, яку частину періоду на виході буде логічна одиниця. Недоліками є відносна складність реалізації, яка потребує прописування макросів, а також, те, що цей метод при роботі постійно використовує процесор мікроконтролера, і тому ніякий інший код не може виконуватися одночасно з ним.

# 3 Другий метод – використання вбудованого таймеру та PWM

## 3.1 Принцип роботи таймеру та PWM

Мікроконтролер ATmega328P має 3 вбудованих таймери [2]. Вони зберігають деяке число, яке збільшується кожний тактовий цикл таймеру. Частоту ітерацій таймеру можна виставити рівною тактовій частоті процесора або або від неї. Таймери з номерами 1 та 2 можна використовувати для генерації широтно-імпульсно модульованого (PWM) сигналу. PWM сигнали використовують для того, щоб змоделювати аналоговий сигнал цифровим. На рис. 2.9 [5] зображений приклад, де в провіднику отримується аналоговий струм (червоний колір), хоча до нього прикладають дискретну напругу (синій колір). У мікроконтролері PWM реалізується за допомогою таймеру. В одному з основних режимів роботи на одній з ніжок мікроконтролера встановлюється логічна одиниця, коли таймер

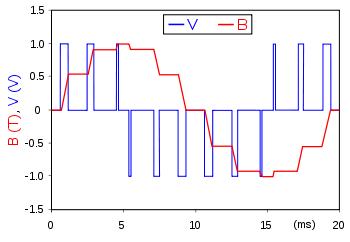


Рис. 3.1

досягає деякого наперед заданого значення, після чого він продовжує рахувати, і коли досягає максимально можливого значення, на ніжку виводиться логічний нуль. Проте цей режим роботи не підходить для поставленої задачі, оскільки при цьому таймер за період робить не менше, ніж 256 ітерацій.

## 3.2 Реалізація

Для отримання найбільшої можливої частоти будемо використовувати таймер номер 1 у режимі роботи CTC (Clear Timer on Compare Match) ([2] 15.9.2). У цьому режимі значення таймеру скидається на нуль, коли він досягає значення записаного в парі регістрів OCR1A (Output Compare Register). При цьому значення на ніжці OC1 (PB1) змінюється з нуля на один і навпаки. Якщо записати в OCR1A нуль, значення таймеру буде скидуватись кожний цикл процесору і на виході отримаємо сигнал з частотою . Код програми зображений на рис. 3.2.

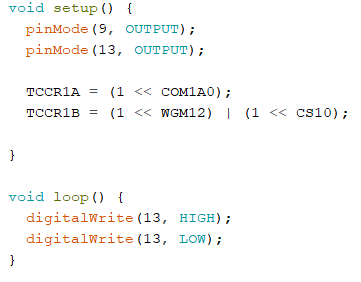


Рис. 3.2

Спочатку настроюємо ніжку 9 Arduino на вихід, бо саме вона відповідає ніжці PB1 [3]. Виставляємо біт COM1A0 в регістрі TCCR1A (Timer/Counter1 Control Register A). Цей біт вмикає режим, при якому при кожному співпадінні значень в таймері і регістрі OCR1A, значення на PB1 змінюється на протилежне ([2] 15.11). Після цього виставляємо біт WGM12 в TCCR1B. Він вмикає потрібний нам режим роботи таймера – CTC. Біт CS10 запускає таймер з максимально можливою частотою роботи, рівною тактовій частоті мікроконтролера. У регістр OCR1A нічого не записуємо, бо при запуску мікроконтролера він ініціалізується нулями. Даний метод отримання найбільшої частоти має перевагу, а саме те, що він не навантажує процесор мікроконтролера, оскільки таймер працює незалежно. Щоб це продемонструвати у головному циклі програми виводиться періодичний сигнал на ніжку 13 Arduino. Схема, використана для симуляції, та її результат, зображені відповідно на рис. 3.3 та рис. 3.4. Можна побачити, що на ніжці PB1 генерується сигнал з частотою .

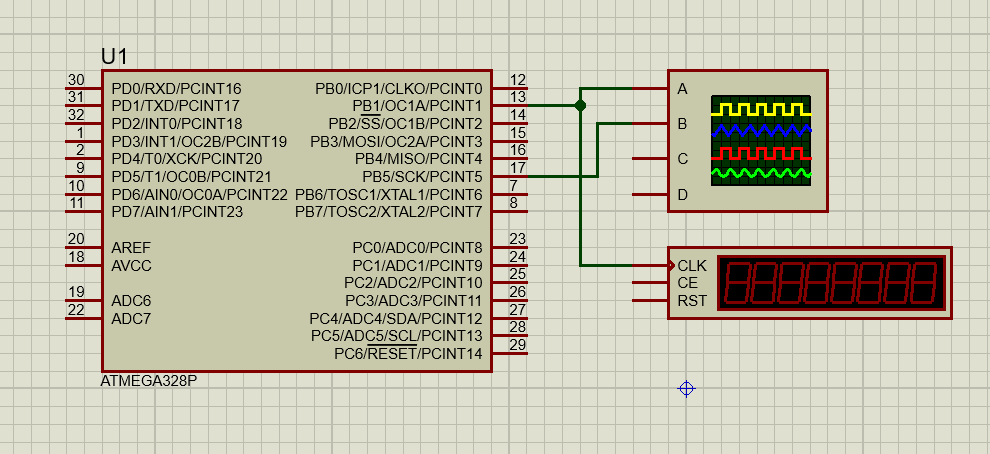


Рис. 3.3

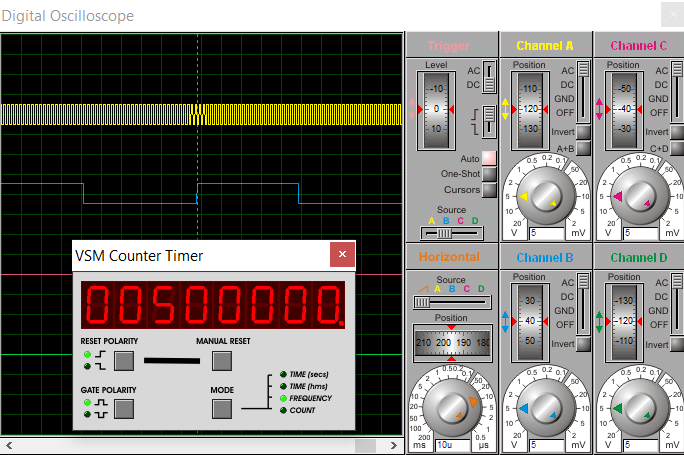


Рис 3.4

Код програми можна легко модифікувати для генерації інших частот, кратних тактовій. Наприклад код на рис 3.5 генерує сигнал з частотою

(рис 3.6).

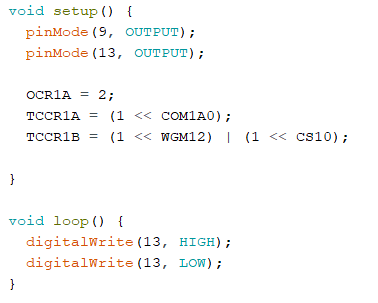


Рис 3.5

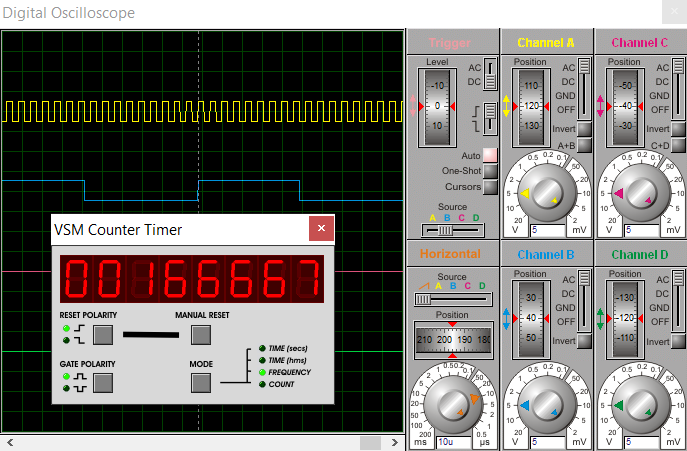


Рис 3.6

## 3.3 Висновки

За допомогою вбудованого таймеру можна отримати максимально можливу теоретичну частоту, як і будь-яку кратну їй. Найбільшою перевагою цього методу є те, що процесор при цьому залишається вільним і може виконувати довільні інструкції, крім тих, що стосуються таймеру номер 1. Недоліком є неможливість отримання сигналів, де логічна одиниця та нуль відображаються різний час за період (як на рис. 2.8). Це можна зробити, але при частоті набагато меншій за максимальну.

# 4 Зміна fuse бітів

Мікроконтролер ATMega328P має так звані fuse біти ([2] 27.2), які можна змінити в процесі програмування. Вони дозволяють налаштовувати деякі параметри мікроконтролера, наприклад, його тактову частоту. З їх допомогою можна вивести тактовий сигнал на одну з ніжок мікроконтролера ([2] 8.9). Проте плата Arduino не дозволяє змінити fuse біти, тому потрібно використовувати зовнішній програматор. В його якості була використана інша плата Arduino Uno, як описано в [6]. Дві плати були з’єднані одна з одною як зображено на рис. 4.1 ([6]), 4.2. В плату, що була програматором, був завантажений скетч “ArduinoISP” з прикладів Arduino IDE. Після цього для комунікації з мікроконтролером використовувалася консольна програма avrdude.

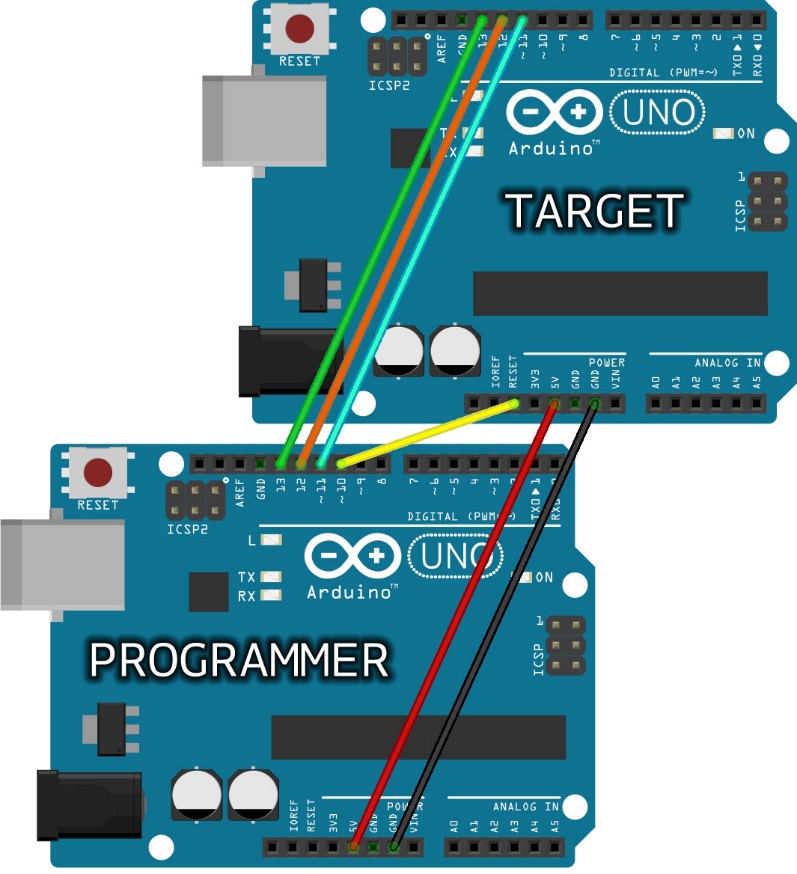


Рис. 4.1

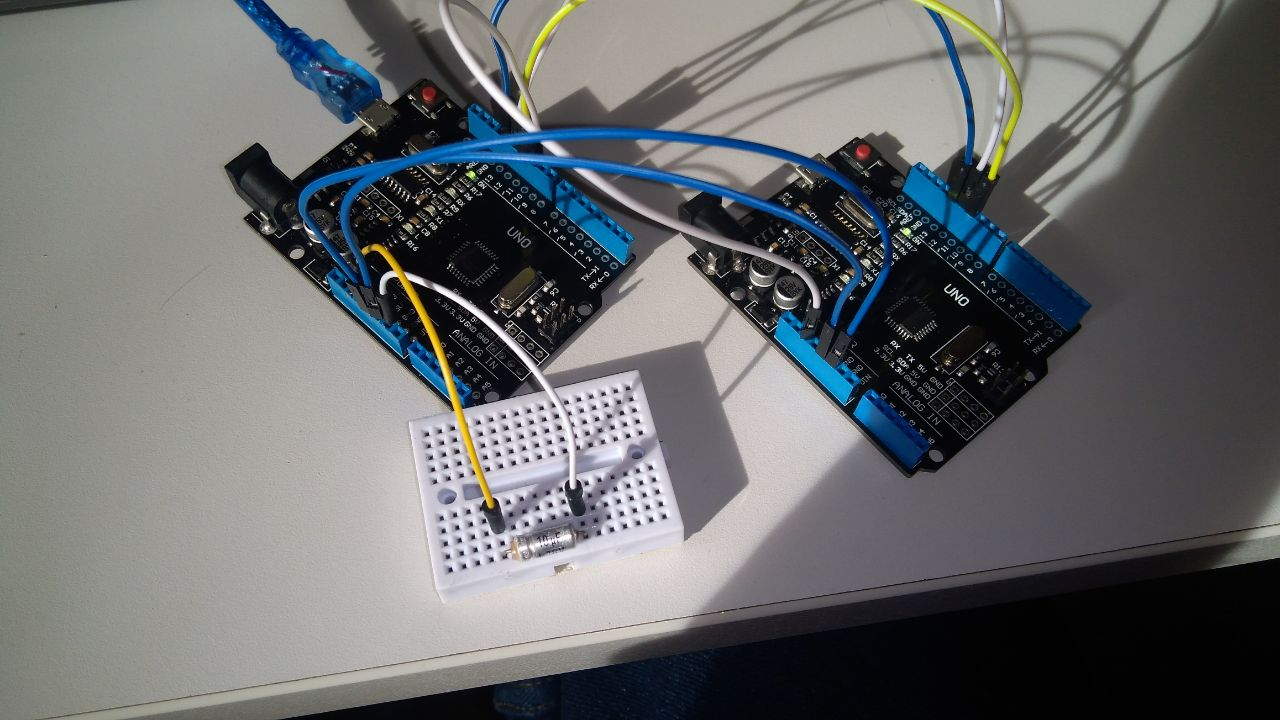


Рис. 4.2

Спочатку були отримані значення “fuse” бітів за допомогою команди

avrdude -c arduino –p m328p –P COM4 -b 19200 -v

яка виводить інформацію про мікроконтролер. Отримані значення fuse бітів показані на рис. 4.3.

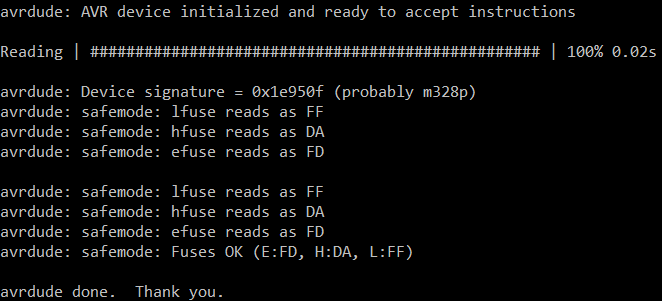


Рис. 4.3

Нас цікавить low fuse byte або lfuse, його значення FF. Це означає, що всі біти рівні 1, тобто не запрограмовані (fuse біт вважається запрограмованим, тобто ввімкнутим, якщо рівний нулю). Згідно з таблицею 27-7 в [2] (рис. 4.4), потрібно запрограмувати біт 6 (clock output) в low fuse byte.

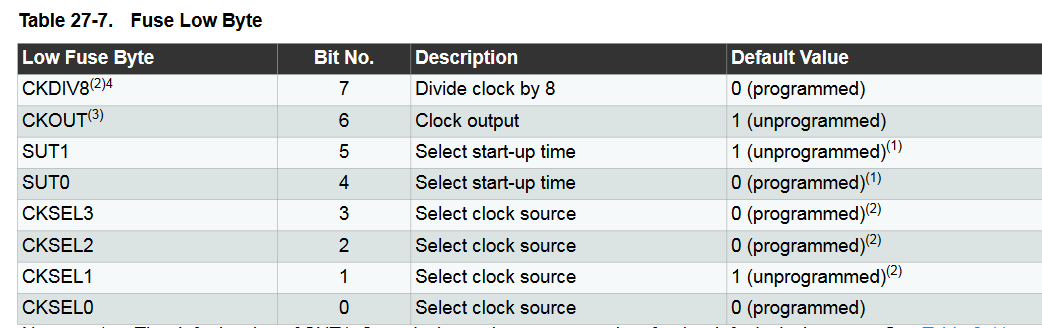


Рис. 4.4

Якщо залишити всі інші біти незапрограмовані (рівні 1), то отримаємо значення байту 10111111 або BF. Програмуємо це значення за допомогою команди

avrdude -c arduino –p m328p –P COM4 -b 19200 -U lfuse:w:0xBF:m

Результат її виконання показаний на рис. 4.5.

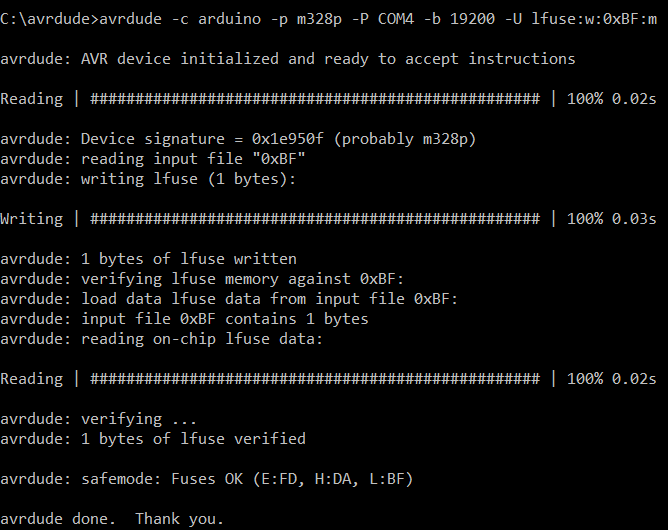


Рис. 4.5

Згідно з 27.2 в [2], тактова частота виводиться на ніжку PB0 мікроконтролера. Згідно з [3], це відповідає ніжці 8 Arduino. Підключивши туди осцилограф, отримуємо сигнал частотою 16 МГц, зображений на рис. 4.6, 4.7. Сигнал спотворений, проте це пов’язано з паразитними індуктивностями та ємностями з’єднувальних провідників, які проявляють себе через високу частоту сигналу. Рис 4.7 отриманий на іншому щупі осцилографа і показує менші спотворення.



Рис. 4.6



Рис. 4.7

Щоб повернути fuse біти до початкове значення використовуємо команду

avrdude -c arduino –p m328p –P COM4 -b 19200 -U lfuse:w:0xFF:m

**ВИСНОВКИ**

У роботі було розглянуто три методи отримання сигналу максимально можливої частоти на платі Arduino Uno Rev3. Перший полягає у повторному виконанні мікроконтролером інструкцій, які виводять на якусь ніжку по черзі то один, то нуль. Його можна точно відрегулювати, проте він завантажує процесор. Другий метод використовує вбудований таймер у режимі порівняння з наперед заданим значенням. Він має перевагу, тому що виконується паралельно до інструкцій основного процесору. Обидва методи видають частоту рівну половині тактової. Порівнюючи ці два методи, можна зробити висновок, що другий метод набагато кращий, бо єдиним його недоліком є можливість генерувати лише одну форму сигналу, де довжина імпульсу та паузи рівні. Проте генерація інших видів сигналів не представляє значний практичний інтерес, і тоді використання таймеру є найкращим методом отримання максимальної частоти за допомогою Arduino. Якщо ж є можливість використовувати зовнішній програматор, то, змінивши fuse біти, можна отримати частоту, рівну тактовій.

**ДЖЕРЕЛА**

1. <https://store.arduino.cc/arduino-uno-rev3>
2. <http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf>
3. <https://www.arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf>
4. <https://github.com/arduino/ArduinoCore-avr/blob/master/cores/arduino/wiring_digital.c>
5. <https://en.wikipedia.org/wiki/Pulse-width_modulation>
6. <https://www.arduino.cc/en/tutorial/arduinoISP>