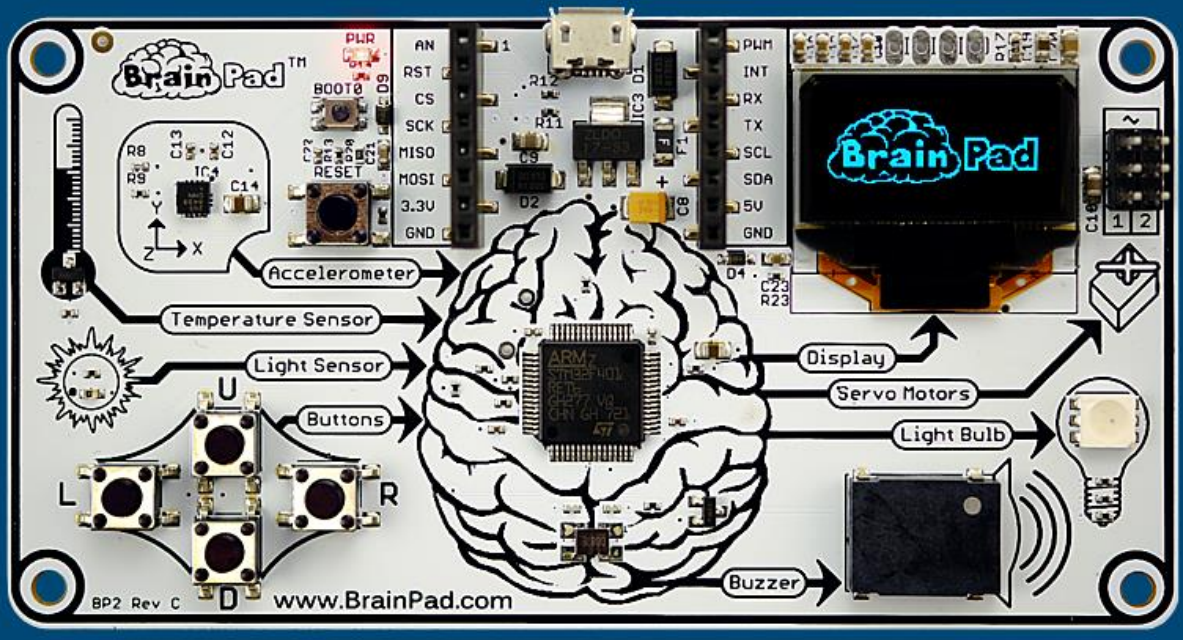
GHI electronics

BrainPad™

BEGINNERS' GUIDE



ЗМІСТ

ВСТУП

BrainPad – потужний STEM інструмент для навчання людей будь-якого віку від маленьких дітей до студентів та професіоналів.

Ця безкоштовна електронна книга – своєрідний гід для новачків у світ BrainPad. Більше матеріалів ви можете знайти тут https://www.brainpad.com.

АВТОР

Гас Icca – засновник і CEO компанії GHI Electronics. Він почав захоплюватись програмуванням та електронікою з дитинства. Спочатку було дуже важко навчатися через обмежений доступ до ресурсів та відсутності інтернету. Ця постійна боротьба за знання привела його на шлях навчання інших людей. Він витратив дуже багато часу, задіявши ресурси своєї компанії, щоб створити доступний спосіб поділитися своїми знаннями. У результаті – створив BrainPad.

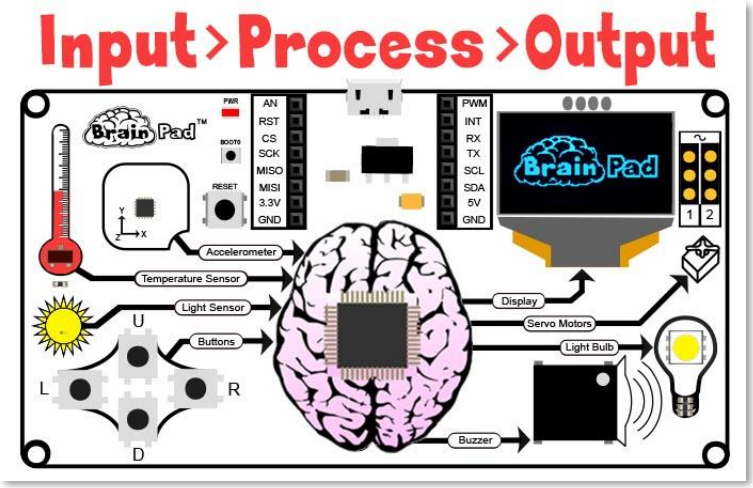
ГЛОБАЛЬНЕ МИСЛЕННЯ

Ми багато працюємо над тим, щоб BrainPad став доступним для кожної культури та мови. Якщо ви маєте труднощі та хочете перекласти цю книгу на свою мову, будь ласка, повідомте нас про це, перейшовши за посиланням https://www.brainpad.com/contact-us.

ЛІЦЕНЗІЯ

КОНЦЕПЦІЯ

BrainPad не потребує довготривалих інструкцій. Він логічно пояснює, як працює комп’ютер. В ньому чотири входи (inputs), які пов’язані з «мозком». «Мозок» (процесор) обдумує інформацію, яка надходить та відображає результат на чотирьох виходах (outputs).



STEM

STEM — це абревіатура від Science (наука), Technology (техніка), Engineering (інженерія) і Mathematics (математика). Вперше поняття STEM було представлене Національним науковим фондом США у 2001 році. Зараз STEM є однією з найбільш обговорюваних тем в освітній системі Сполучених Штатів Америки. Однак викладання STEM — це не лише ці чотири галузі в освіті. Освітній підхід STEM спрямований на те, щоби зв'язати навчання у класі з реальним світом, підкреслюючи комунікацію, співпрацю, критичне мислення і творчий підхід у процесі викладання інженерного проектування.

ФІЛОСОФІЯ

Освітній системі STEM потрібні не іграшки, а платформи, які розвивають. Якщо ви розвиваєтесь, значить і BrainPad з вами! Під егідою інструментів STEM існує багато «іграшок», але більшості з них не вистачає багатогранності, щоб підтримувати цікавість студентів на тривалий час. BrainPad включає в себе програмування (від новачка до професіонала), електроніку, інженерну справу, робототехніку та багато іншого на недорогій платформі, яку можуть використовувати як школярі, так і студенти. Це підтверджується 15-річним досвідом використання цієї платформи в США та країнах Європи. Ті ж інструменти, які використовують комерційні клієнти для роботи з промисловим контролерами, можуть бути використані і для роботи з BrainPad. BrainPad приводить студентів до програмування шляхом побудови програм за допомогою перетягання блоків до професійного програмування та інженерії. Жодна інша платформа STEM не здатна на таке.

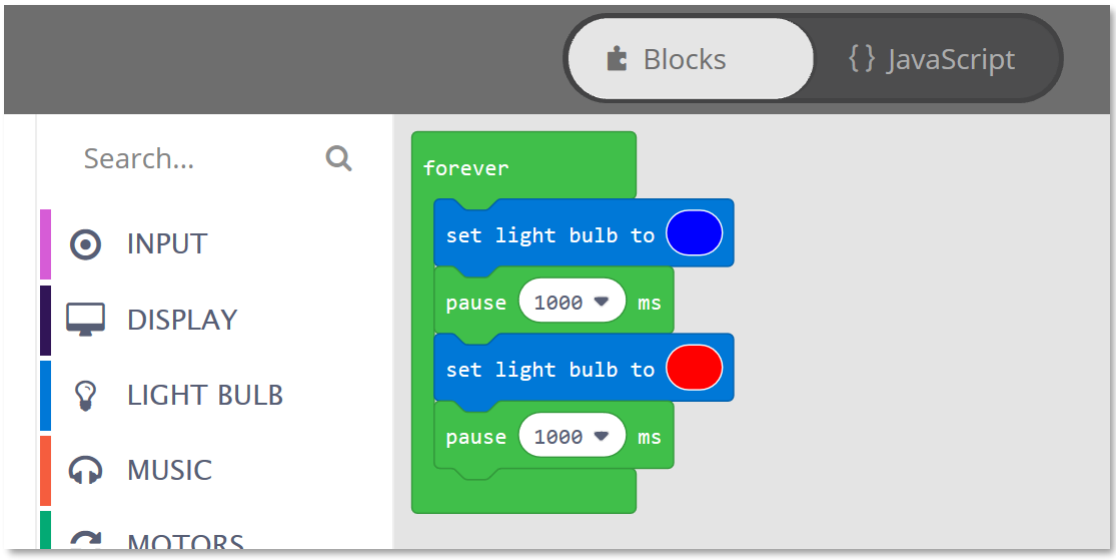
ДВА ШЛЯХИ

BrainPad пропонує два шляхи вивчення програмування: **Start Making (Почни програмувати) і Go Beyond (Виходь за межі).** Перший шлях досить простий, з низьким порогом входження, а другий пропонує вийти за межі і почати використовувати ті є інструменти, що й професіонали.

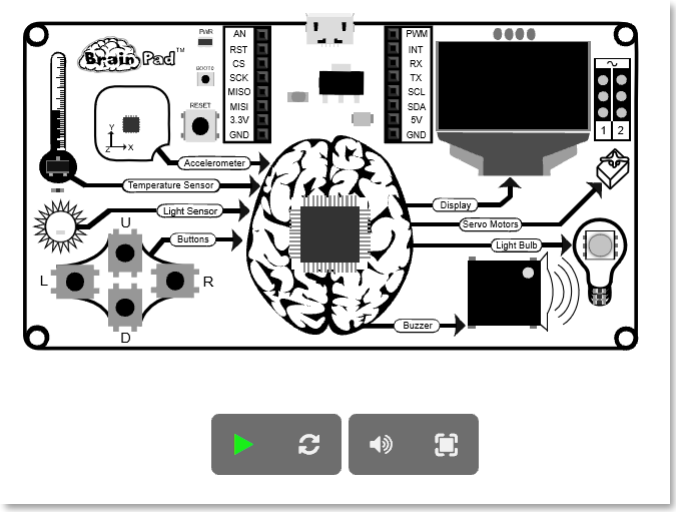
START MAKING (ПОЧНИ ПРОГРАМУВАТИ)

Починаючи роботу зі **Start Making** вам не доведеться нічого встановлювати на комп'ютер – усі інструменти вже встановлені і готові до роботи через інтернет-браузер, використовуючи Microsoft MakeCode (графічний редактор коду). Другий шлях – **Go Beyond** потребує встановлення і налаштування програмного забезпечення. BrainPad працює в середовищі Microsoft Visual Studio.

З Microsoft MakeCode (Start Making) ви можете програмувати, просто складаючи блоки або використовуючи мову програмування JavaScript прямо у вашому інтернет-браузері. На малюнку, що нижче, наведений приклад програми, яка кожну секунду включає спочатку синій світлодіод, а потім червоний у нескінченному циклі.



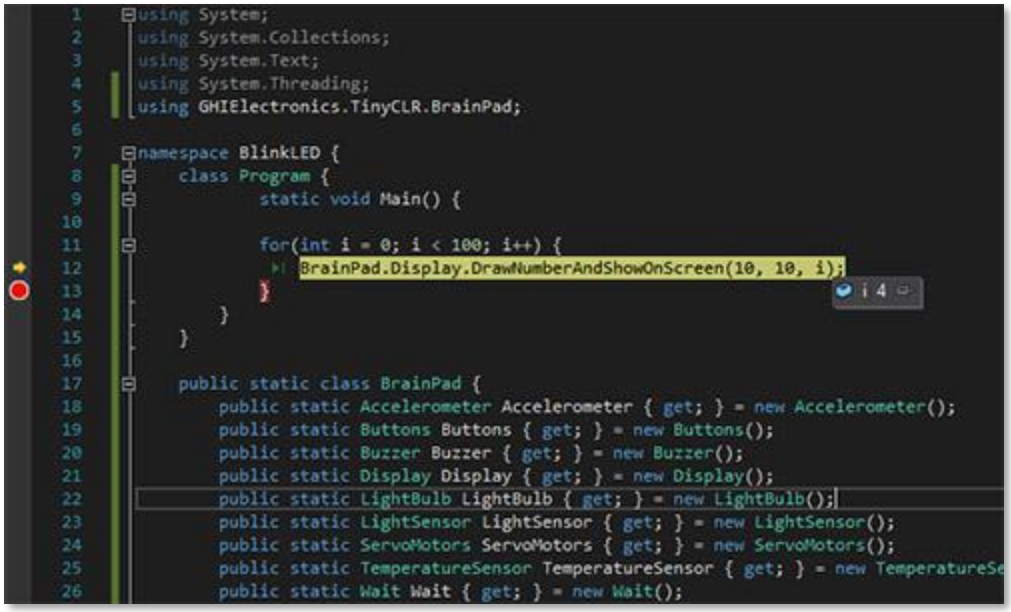
На цьому малюнку – скріншот симулятора, тобто, віртуальний BrainPad. Він запускає програму прямісінько на дисплеї вашого монітору. Знаходиться він у лівому боці екрану у вікні Microsoft MakeCode. Симулятор дозволяє вам тестувати програми прямо у вашому браузері без завантаження в BrainPad. Це чудова можливість почати роботу вже зараз, навіть якщо ви ще не маєте BrainPad.



GO BEYOND (ВИХОДЬ ЗА МЕЖІ)

Йдучи шляхом Go Beyond, можливості у вас набагато серйозніші! Вам потрібно встановити на комп'ютері Microsoft Visual Studio, щоб програмувати і налагоджувати програми, запущені на BrainPad.

Завдяки операційній системі TinyCLR OS™ від GHI Electronics у вас є можливість програмувати BrainPad у середовищі .NET на мовах програмування C# і Visual Basic. Доступна можливість повного налагодження програми: покроковий запуск коду, контрольні точки й інспекція змінних під час запуску тощо. Ви будете писати програми для BrainPad точно так же, як це роблять професіонали, наприклад, для ПК чи мобільних пристроїв. Ця книга опише обидва шляхи.



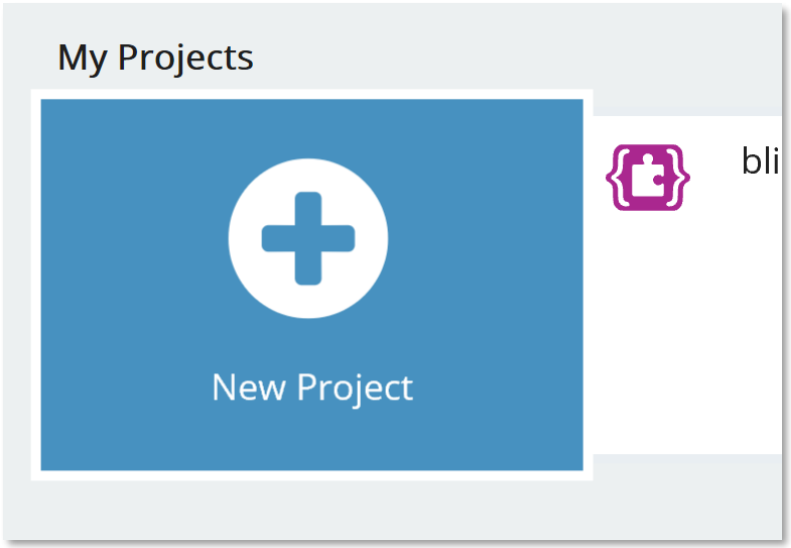
```
using System;
using System.Collections;
using System.Text;
using System.Threading;
using GHIElectronics.TinyCLR.BrainPad;

namespace BlinkLED {
    class Program {
        static void Main() {

            for(int i = 0; i < 100; i++) {
                BrainPad.Display.DrawNumberAndShowOnScreen(10, 10, i);
            }
        }
    }

    public static class BrainPad {
        public static Accelerometer Accelerometer { get; } = new Accelerometer();
        public static Buttons Buttons { get; } = new Buttons();
        public static Buzzer Buzzer { get; } = new Buzzer();
        public static Display Display { get; } = new Display();
        public static LightBulb LightBulb { get; } = new LightBulb();
        public static LightSensor LightSensor { get; } = new LightSensor();
        public static ServoMotors ServoMotors { get; } = new ServoMotors();
        public static TemperatureSensor TemperatureSensor { get; } = new TemperatureSe
        public static Wait Wait { get; } = new Wait();
```

ЙДЕМО ЩЕ ДАЛІ!

Для BrainPad можна також писати програми, використовуючи інші мови програмування й інструменти, не описані у цій книзі. Наприклад, середовище розробки Arduino https://www.arduino.cc, MicroPython https://www.micropython.org і Arm Mbed https://www.mbed.com.

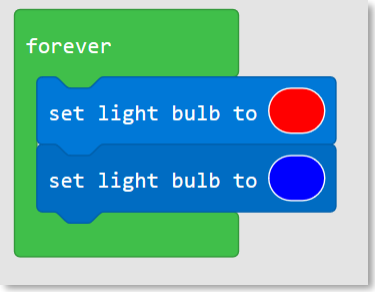
START MAKING (ПОЧНИ ПРОГРАМУВАТИ)

Досить розмовляти – починаймо працювати і отримувати задоволення! Ми будемо використовувати Microsoft MakeCode, тож переходимо за посиланням https://makecode.brainpad.com/. На цьому сайті ви знайдете багато цікавих матеріалів та проектів. Але ми починаємо з нового. Просто натисніть кнопку New Project.



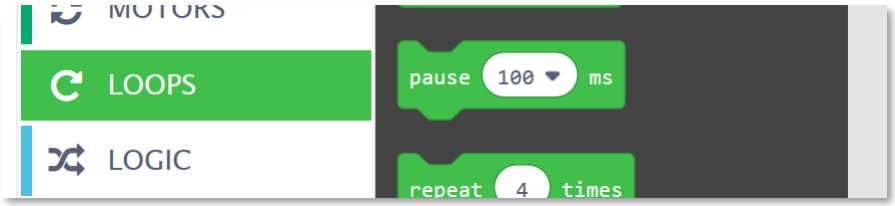
Зі вкладки LIGHT BULB у головному меню перетягніть блок «set light bulb to» і помістіть його у блок «forever. Блок «forever» використовуємо, коли ви хочете, щоб ваша програма виконувалась весь час. Усі команди всередині цього блоку BrainPad буде виконувати, поки він ввімкнений. І коли черга інструкцій буде виконана, BrainPad запустить код знову, починаючи від першої інструкції у блоці «forever». Це називається нескінченним циклом і часто використовується у програмуванні.

Повторіть цей крок, але змініть колір у другому блоці, натиснувши на кольоровий овал. Ось, що у вас повинно вийти:

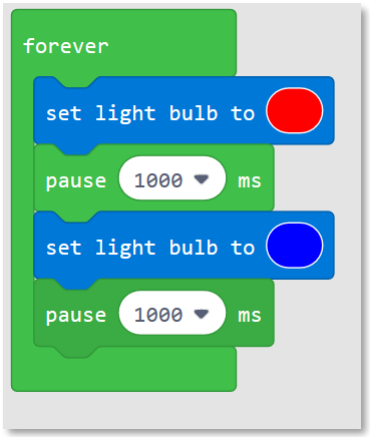


Програма, яку ми створили, повинна змінювати колір світлодіоду на червоний, а потім знову на синій миттєво і нескінченно. Якщо ви подивитесь на симулятор, то помітите, що світлодіод буде працювати не так, як ми очікуємо. І коли ви завантажите програму на BrainPad, то впевнитеся, що програма працює неправильно. Але чому? Справа у тому, що ми не вказали, як довго треба зачекати світлодіоду перед тим, як змінити свій колір. Зараз колір змінюється, але настільки швидко, що око не встигає це уловити.

З меню LOOPS перетягніть блок «pause».



Нам потрібно виставити паузу після ввімкнення червоного і синього світлодіодів. Також давайте змінимо час паузи на одну секунду.



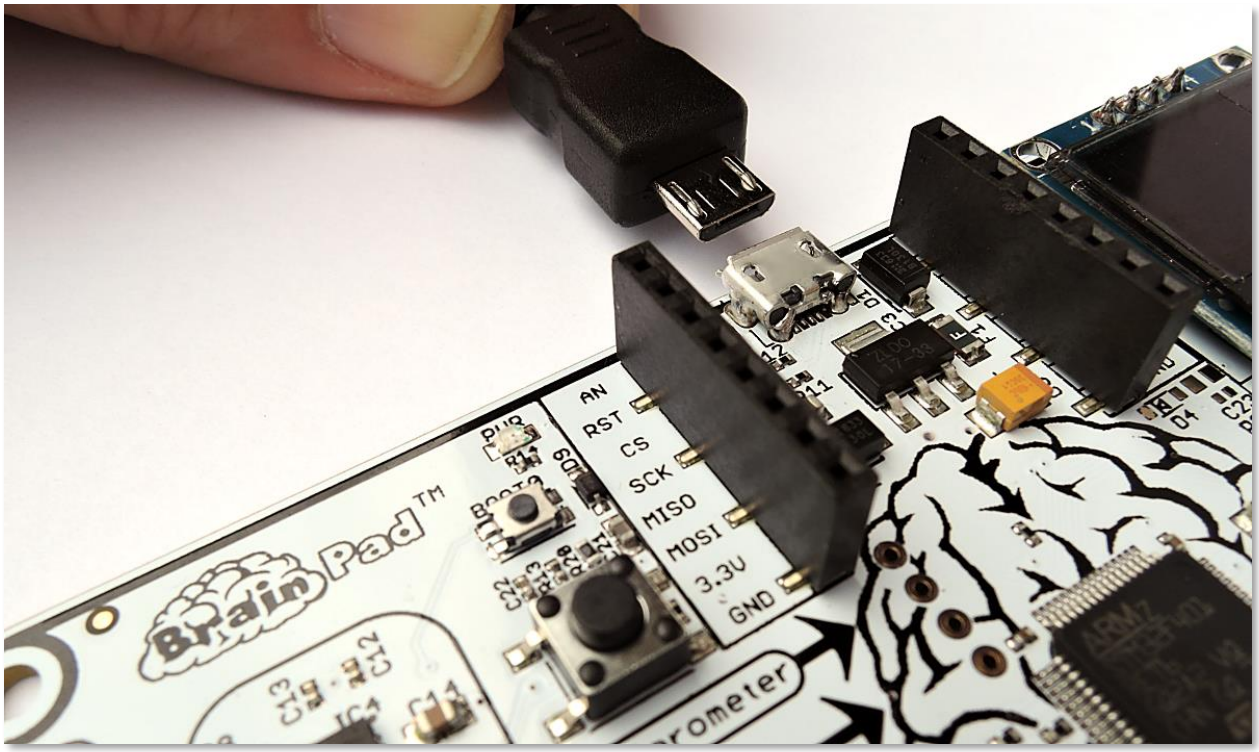
Подивіться на симулятор і впевніться, що світлодіоди працюють так, як ми очікували. Непогано, правда ж? А якщо ми завантажимо нашу програму на BrainPad?

СКОПІЮВАТИ НА BRAINPAD

Процес копіювання програми на BrainPad доволі простий, але на початку може здатися трохи складним. Спочатку під'єднайте BrainPad до вашого комп'ютера, використовуючи мікро-USB кабель. До речі, ви можете використовувати Windows, Chromebook, Mac чи будь-який інший сучасний прилад, в якому є інтернет-браузер і USB-порт.

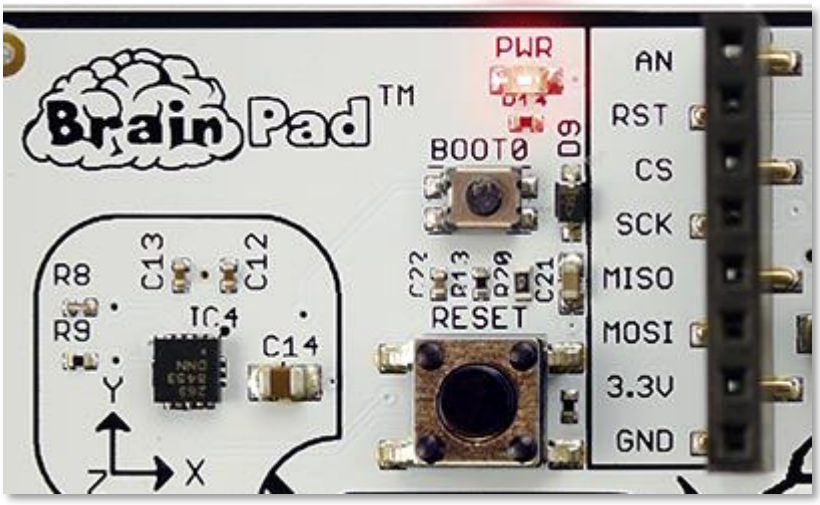
Мікро-USB кабель є в комплекті вашого BrainPady. Хоча, можливо, у вас вже є такий. Якщо раптом немає, то цей тип кабелю дуже поширений і його можна купити будь-де за доступною ціною.

Підключіть маленький коннектор до BrainPad.

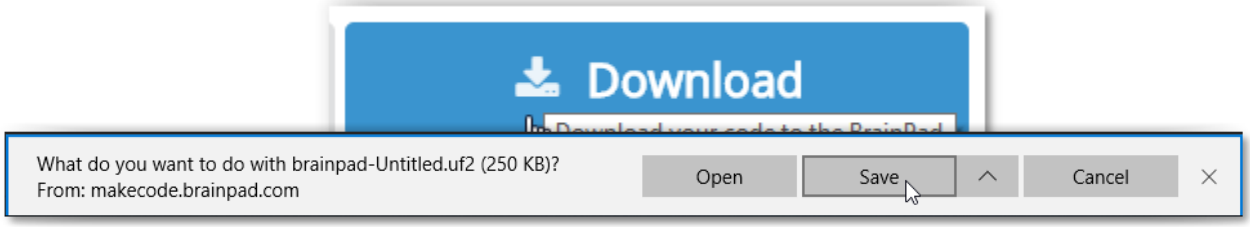


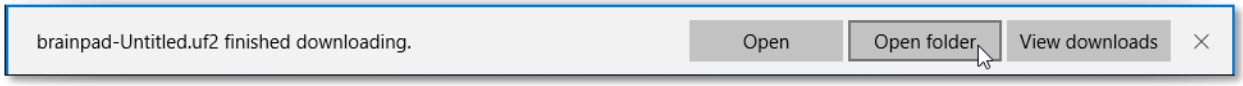
А великий коннектор – до комп’ютера.

Коли BrainPad підключиться до вашого комп’ютера, на ньому загориться червоний світлодіод.



Повернімося у браузер з програмою, яку ми створили раніше. Натисніть на кнопку «зберегти» (save) і збережіть файл на комп’ютер.

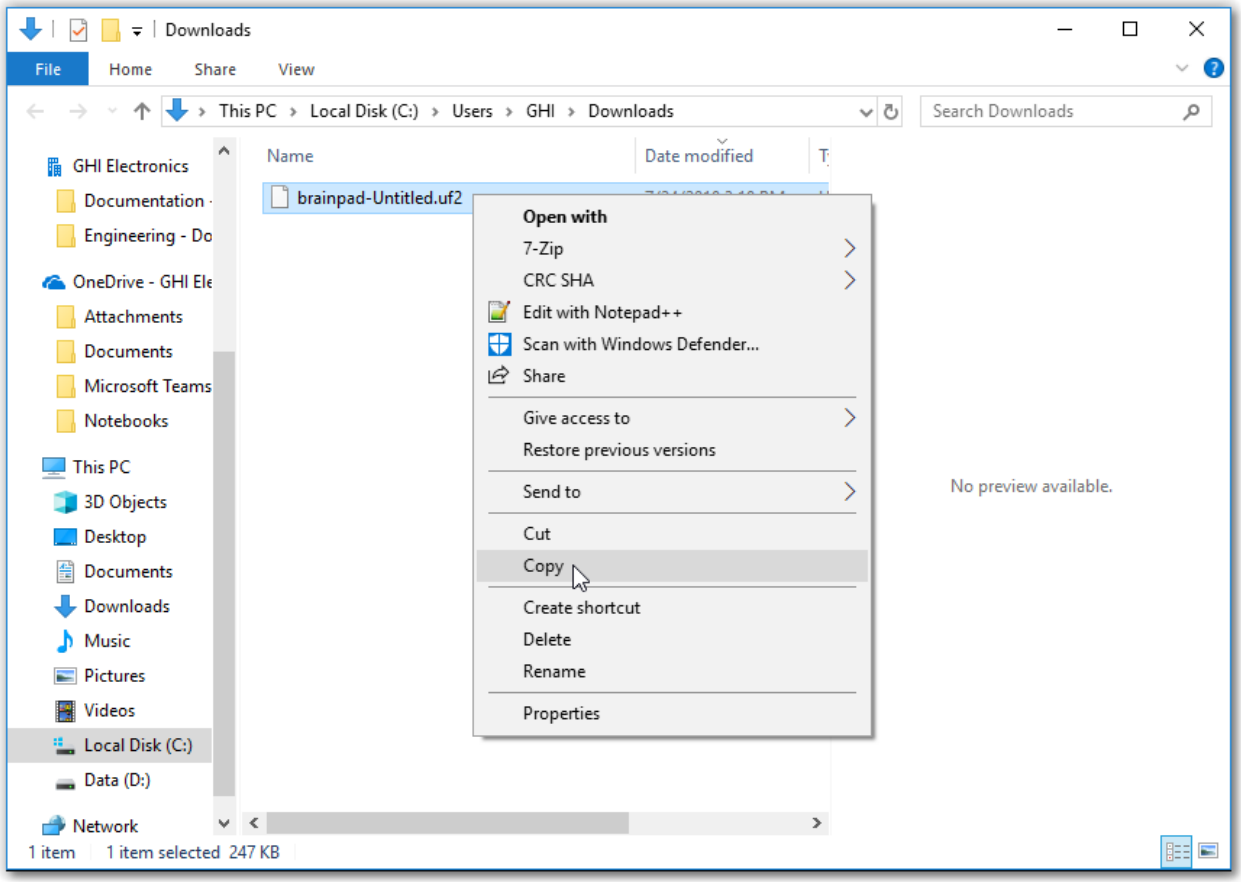




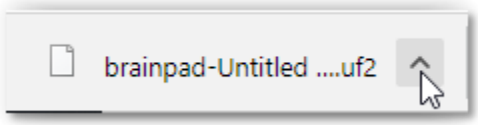
Якщо ви використовуєте Microsoft Edge, то побачите діалогове вікно як на малюнку нижче:

Натисніть кнопку «зберегти», і далі – «відкрити папку» (Open folder) в наступному вікні, щоб побачити збережений файл.

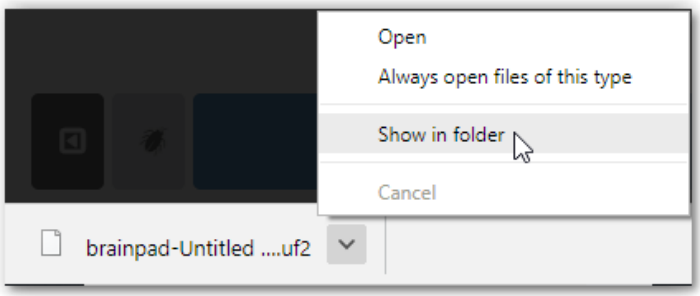
Файл буде підсвіченим. Ви можете натиснути праву кнопку мишки і вибрати «копіювати» (Copy).



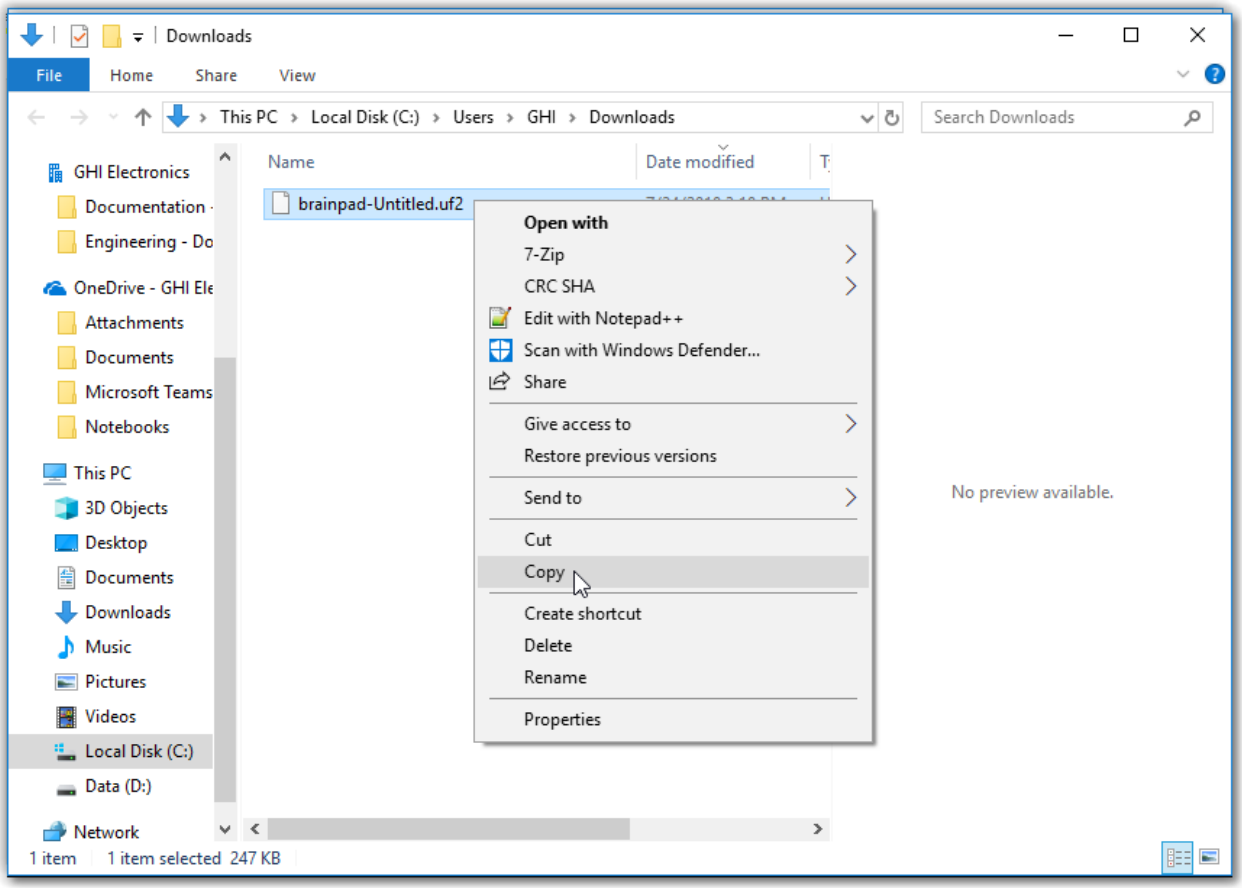
Якщо ви використовуєте Chrome, файл буде доступний у нижньому лівому куті екрану:



Натисніть на маленьку стрілку «вверх» і оберіть «Показати у папці» (Show in folder).

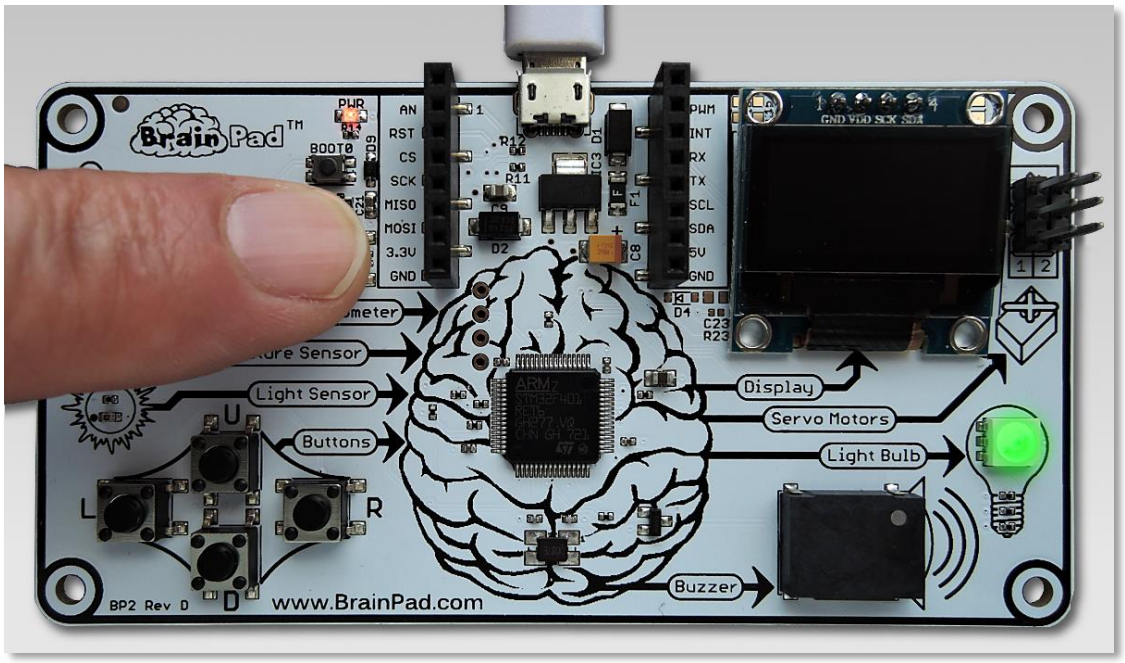


Завантажений файл буде підсвічений і ви зможете натиснути праву кнопку миші і обрати «Копіювати» (Copy)

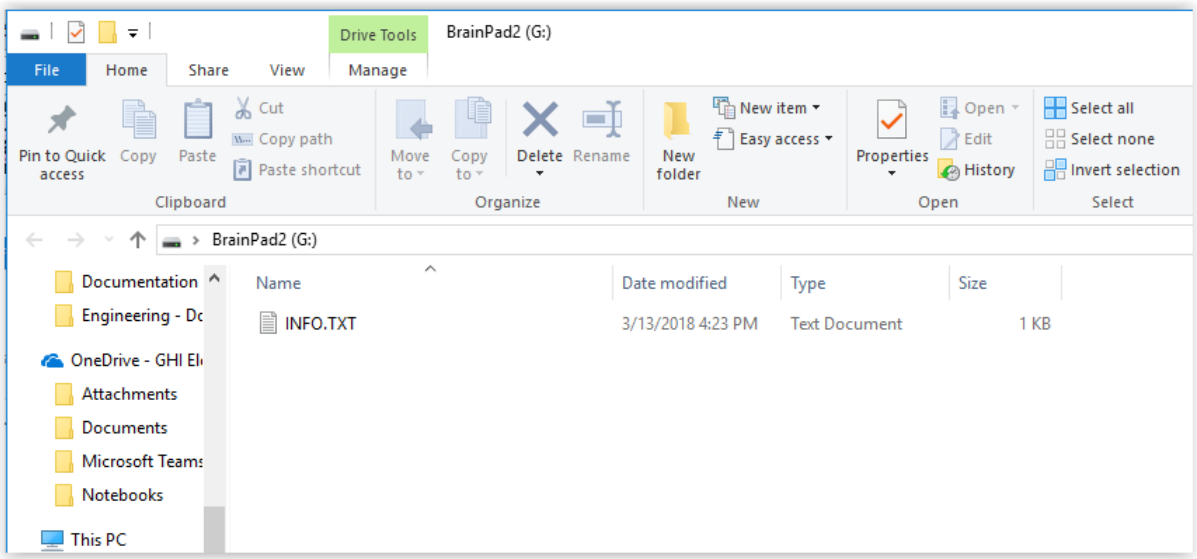


Працюючи у будь-якому браузері, Microsoft MakeCode не видаляє ваші старі файли. Якщо ви завантажили одну і ту ж програму кілька разів, до назви файлу буде додаватися номер (наприклад, «brainpad-Untitled (1).uf2») . Впевніться, що ви скопіювали останній завантажений файл. Останній файл буде підсвіченим. У його назві також буде найбільший номер в дужках і остання дата і час його зберігання.

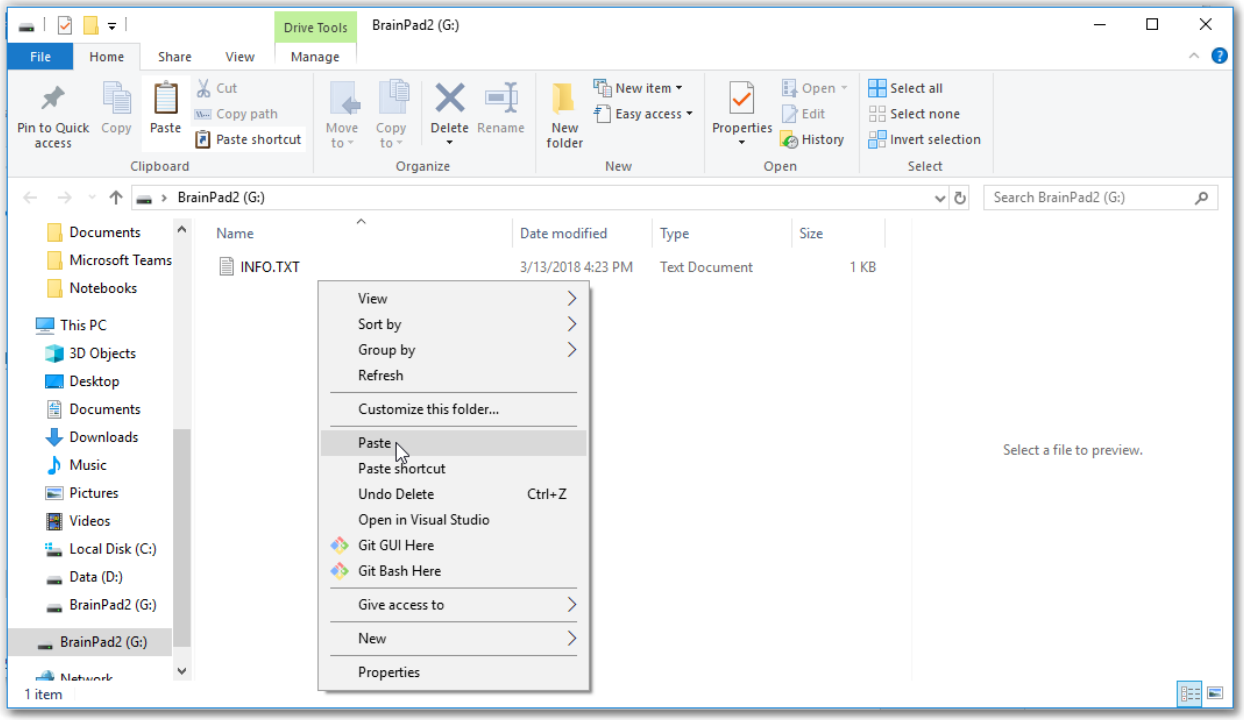
Тепер, коли BrainPad підключений до вашого комп’ютера, на ньому повинен горіти червоний світлодіод. Натисніть і утримуйте кнопку «reset» близько трьох секунд. Повинен загорітися зелений світлодіод.



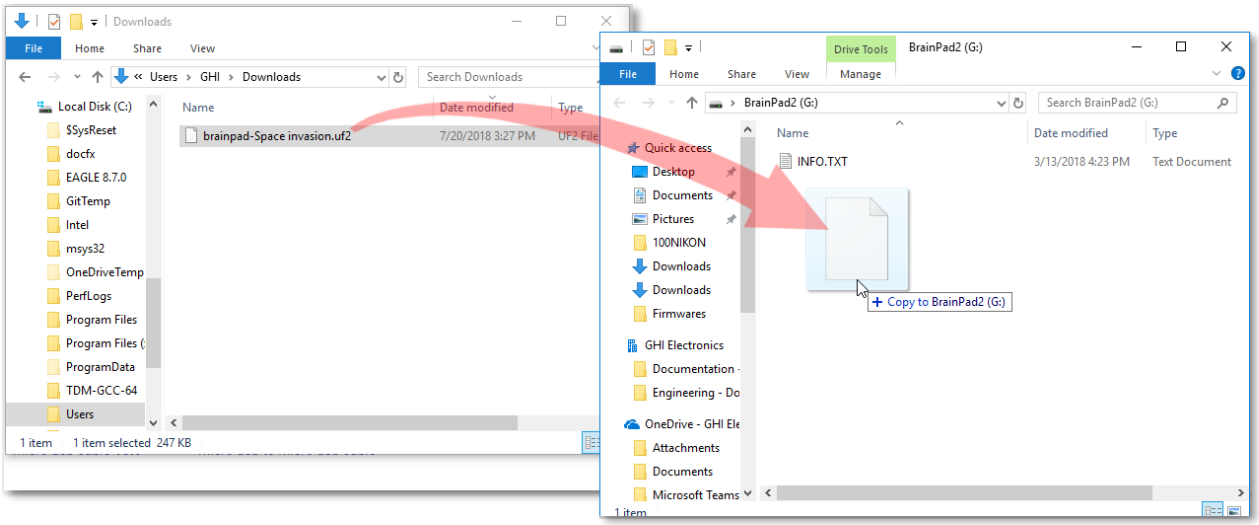
Ваш комп’ютер розпізнає USB-накопичувач. Це може тривати до хвилини. Потім на вашому екрані з’явиться вікно з назвою «BrainPad».

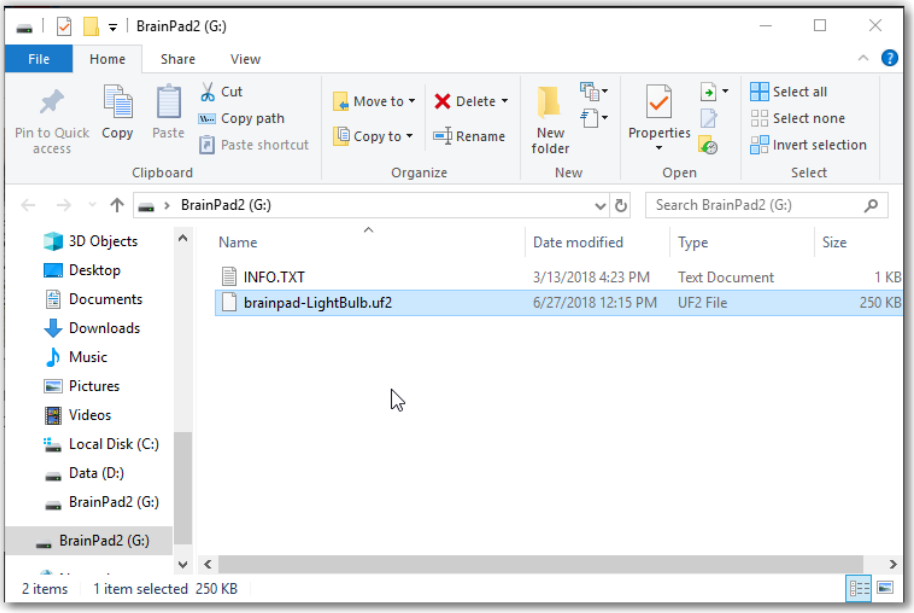


У цьому вікні відображається те, що знаходиться у пам’яті BrainPad. Зараз натисніть праву кнопку миші і оберіть «вставити» (paste), щоб скопіювати файл на BrainPad.



Також ви можете просто перетягнути завантажений файл у вікно BrainPad.



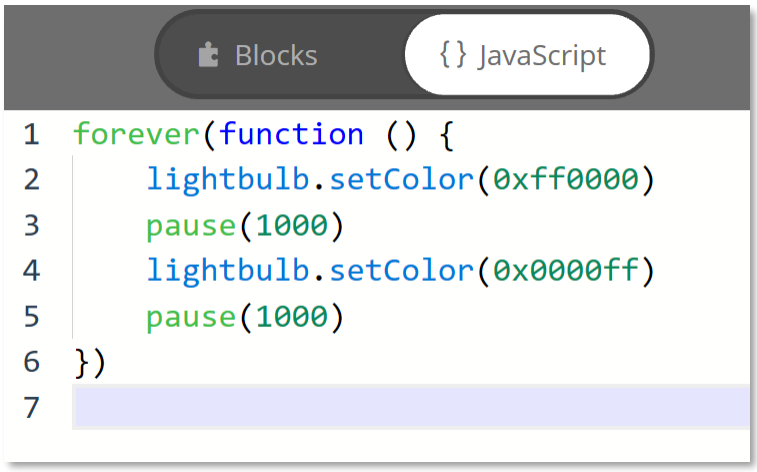


Як тільки ви це зробите, світлодіод почне мерехтіти і після цього BrainPad запустить завантажену програму.

JAVASCRIPT

Блоки – дуже захоплююча робота для початку… Але ваші програми ставатимуть більшими і вам доведеться набратися сміливості й почати писати код. Не хвилюйтеся, ви зможете це зробити, коли будете готові. Однією з властивостей Microsoft MakeCode є переклад блоків на мову програмування JavaScript, і навпаки – з JavaScript у блоки.

Повернімось до програми, яку ми написали, і натиснемо на кнопку JavaScript.



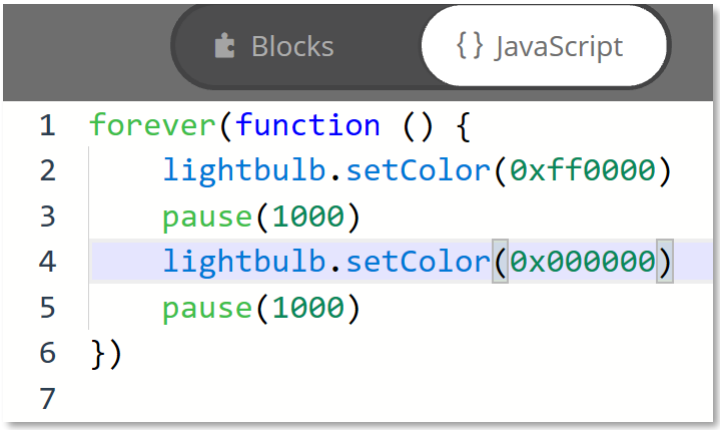
Погляньте на цей код. Ви знаходите схожість між блоками і кодом? Один момент, який може бути незрозумілим – це кольори, для визначення яких JavaScript використовує число в HEX форматі. Інший код зрозумілий?

Представлення чисел у HEX форматі базується на шіснадцятковій системі чисел замість звичної для нас десяткової. Цифри у шіснадцятковій системі представлені від 0 до F, на відміну від десяткової, в якій від 0 до 9 (числа, які ми використовуємо у житті). Якщо ми спробуємо порахувати від 0 до 16 у шіснадцятковій системі, то це виглядатиме ось так: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10.

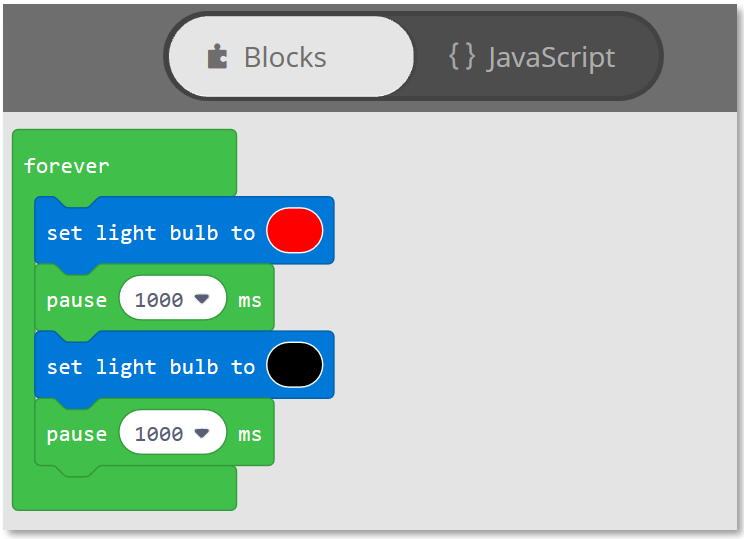
"0x" означає початок шіснадцяткового числа і дає нам зрозуміти, що ми дійсно маємо справу з шіснадцятковим числом. Часто використовують скорочення HEX для позначення шіснадцяткових чисел. Обов'язково треба згадати про бінарну систему чисел, або двійкову, як її ще називають. Тут використовуються тільки 0 і 1. Отож, тепер ми знаємо, що числа можуть бути представлені двійковою, десятковою і шіснадцятковою системою обчислення. Це трохи може заплутати, але не хвилюйтесь – через деякий час ви будете легко у цьому орієнтуватися.

Коли використовують шіснадцяткове число для позначення кольору, перші дві цифри означають присутність червоного у кольорі, наступні дві – присутність зеленого, треті – синього. "00" означає, що інтенсивність кольору дорівнює нулю або колір відключений. "FF" означає повну інтенсивність кольору. Яскравий червоний позначається як 0xFF0000, білий – 0xFFFFFF.

Що станеться, якщо ми змінимо один з блоків «setColor» на 0x000000? Давайте спробуємо.



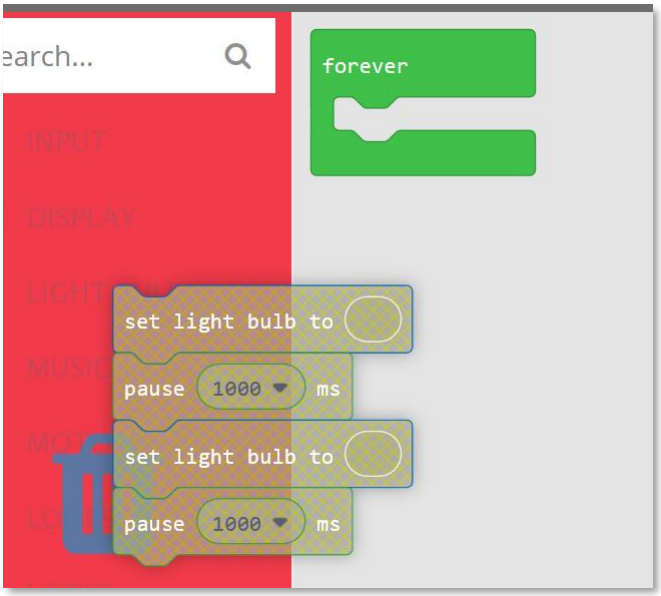
Переключаймося назад на блоки і дивимося, що змінилося.



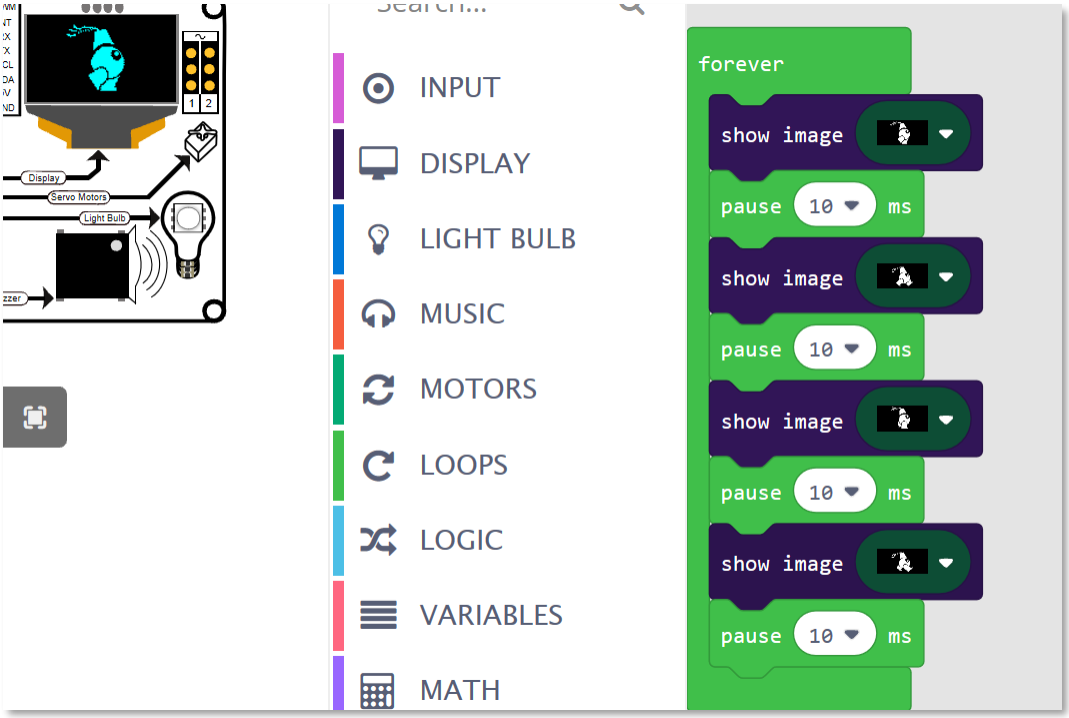
Схоже, що колір змінився з синього на чорний. Ви, напевне, здогадуєтесь, що встановивши на світлодіоді чорний колір, ми його тим самим виключили. Ви можете побачити це на симуляторі. Світлодіод повинен на секунду загорітися червоним, а потім виключитися на секунду.

ЦІКАВІШЕ З ДИСПЛЕЄМ

Якщо вам здається, що мерехтіти світлодіодом – це цікаво, то починайте використовувати дисплей! Якщо ви все ще використовуєте блоки з попереднього прикладу, перетягніть їх назад в меню, щоб видалити.



Якщо ви випадково видалили блок «Forever», ви легко можете його відновити з меню LOOPS. З меню DISPLAY перетягніть блок «show image» і оберіть перший з чотирьох малюнків «людина, яка йде». Додайте інші три зображення і пам'ятайте про паузу в 10 мілісекунд між ними. У блоці «пауза» немає опції 10 мілісекунд, але ви можете зробити це самі, просто змінивши число в овалі на 10.

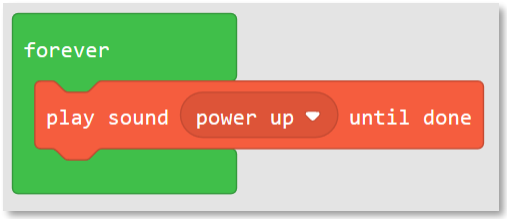


Тепер людина повинна ходити у симуляторі на дисплеї вашого комп'ютера. Рухаймося далі і завантажимо це в наш BrainPad.

ГРАЄМО ПО НОТАХ

У меню MUSIC ви можете знайти окремі ноти…

…чи мелодії.

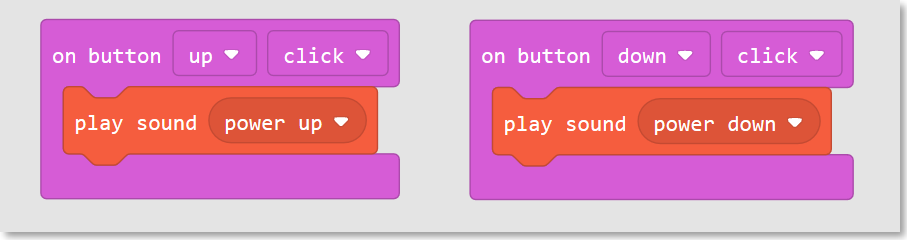


Зверніть увагу як ми використовуємо «play sound until done» блок. Цей блок зупиняє програму, поки не закінчиться програвання мелодії. Блок «play sound» (без «until done») починає виконання наступного блоку у той час, коли мелодія все ще грає. Якщо ви спробуєте обидва блоки, то побачите, що результат дуже відрізняється.

ВИКОРИСТОВУЄМО КНОПКИ

Є два шляхи використання кнопки. Перший – просто перевіряти, чи натиснута кнопка і обрати потрібну дію, якщо вона натиснута. Недолік цього методу полягає втому, що вам постійно треба перевіряти, чи не пропустили ви натиснення. Другий шлях – дозволити системі робити це за вас і в разі «події» натиснутої кнопки автоматично щось робити.

Спочатку спробуємо «читати» стан кнопки, використовуючи події. Наприклад, щоби грала мелодія «power up», коли натиснута кнопка «вверх» і мелодія «power down», коли натиснута кнопка «вниз».



Ви звернули увагу, що ми не використовуємо блок «forever» у цьому прикладі? У вас все ще є цей блок у вашому коді, але у цьому прикладі він не потрібен. Це тому, що система обробляє «події» автоматично, коли кнопка натиснута.

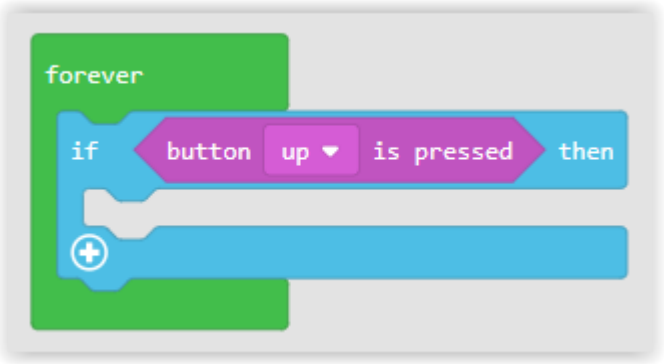
Ці ж блоки у JavaScript коді:

```
input.buttonU.onEvent(ButtonEvent.Click, function () {
    music.playSound(music.sounds(Sounds.PowerUp))
})

input.buttonD.onEvent(ButtonEvent.Click, function () {
    music.playSound(music.sounds(Sounds.PowerDown))
})
```

Ми напишемо програму, яка буде робити те ж саме, але без використання «подій». Щоб перевірити, чи натиснута кнопка, ми будемо використовувати блок «if». Слово «if» у перекладі з англійської означає «якщо». Тобто, «якщо» натиснута, то ми виконуємо якусь дію. Якщо ви знайомі з англійською мовою, то могли помітити, що практично всі імена-інструкції у блоках і JavaScript-код відповідають тим діям, які ми хочемо виконати. «If» ми можемо знайти у меню LOGIC. Перетягніть його у наш вічний цикл.

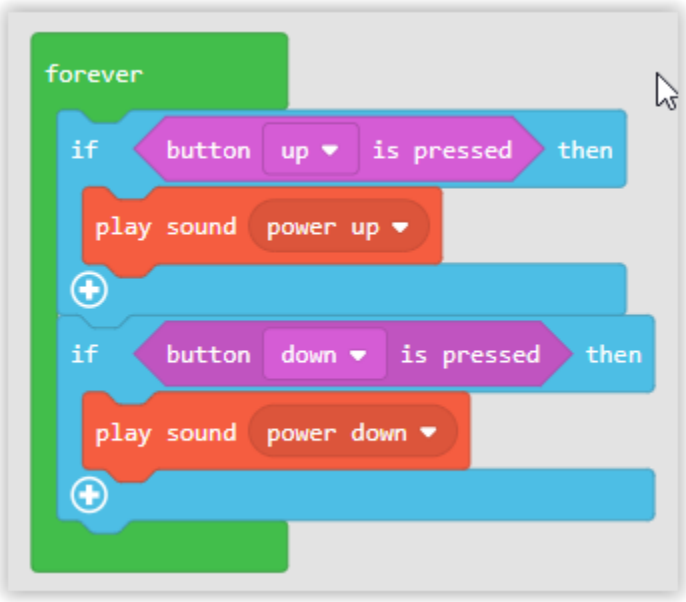
«If» перевіряє стан чогось і це може бути «так» або «ні», а точніше – «true» або «false». За замовчуванням стоїть «true», але ми хочемо замінити цю умову, щоб перевірити кнопку.

Перетягніть «button. . . is pressed» з меню «input» в «if. . . then» як показано на малюнку нижче. Зверніть увагу, що це не подія «on button», яку ми використовували раніше.



У прикладі вище блоки будуть перевіряти кнопку «завжди». Якщо кнопка «вверх» буде натиснута, програма виконає всі команди, які знаходяться всередині блоку «if».

Додаймо більше блоків, щоб зробити нашу програму такою, як показано нижче:

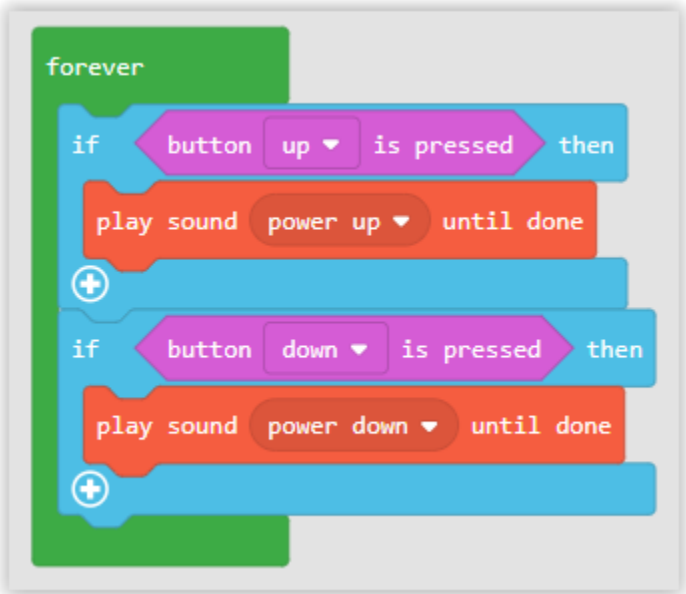


Коли ви запустите цей код, то помітите, що мелодія не програється правильно, поки ви не відпустите кнопку. Причина в тому, що кожен раз, коли BrainPad визначає, що кнопка «вверх» або «вниз» натиснута, він починає програвати нову мелодію, навіть якщо попередня не закінчилася. BrainPad почне програвати мелодію багато разів кожну секунду поки кнопка натиснута.

Існує два шляхи вирішення проблеми. Перший – замінити «button. . . is pressed» блок на «button. . . was pressed» блок. Умова «button. . . was pressed» встановить значення «true», якщо кнопка була натиснута хоча б один раз з моменту останньої перевірки кнопки. Навіть якщо кнопка була натиснута багато разів, буде зареєстроване лише одне натискання. Натискання кнопки «вверх»

або «вниз» під час програвання мелодії обірве її, але нова мелодія не буде повторюватися спочатку, якщо кнопка затиснута.

Другий шлях – замінити блок «play sound. . .» на «play sound. . . until done». Цей блок зупинить виконання іншої частини програми, поки мелодія не закінчиться. Тобто, програма зупиняє перевірку натиснення кнопок або програвання решти мелодій, поки поточна мелодія не буде відтворена. Цей блок не дасть програмі можливість перервати мелодію. Блоки показані нижче.

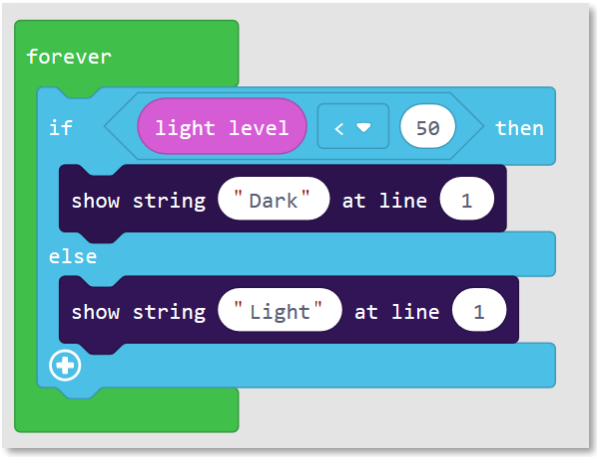


ТУТ ТЕМНО?

Так же, як і кнопки ви можете «читати» рівень освітлення, або ж скористатися «подією» і запускати якусь дію, коли темно чи світло. Однак, читання кнопки і рівня освітленості майже не відрізняються.

Кнопка може бути натиснутою або ненатиснутою. Якщо виникне запитання «Кнопка натиснута?», ви відповісте «Так» - якщо це дійсно так, або «ні» - коли кнопка не натиснута. Датчик світла дає рівень освітленості як значення. Програмісти називають це аналоговим значенням. Давайте «прочитаємо» рівень освітленості і виведемо його на дисплей. Це потрібно робити у циклі «forever», тобто постійно читати освітленість і оновлювати дисплей.

Так як ви тепер експерти у використанні оператора «if», чи вдасться вам вивести на дисплей «темно» чи «світло» у залежності від значення датчика освітлення? Це може бути трохи складно, але пояснимо, як це робити. Оператор «if» потребує чогось, що дає «true» або «false», як наприклад, кнопка. Рівень освітлення - це не «true» або «false», це може бути одне з багатьох значень. Якщо подумати над цим, то немає сенсу у тому, щоб сказати: «якщо світло, то зроби щось». Нам варто сказати «якщо світліше чи темніше, ніж якесь значення, тоді зроби щось». Відкрийте меню LOGIC і додайте блок порівняння, тобто блок «if» плюс блок «else». Подивіться на це:



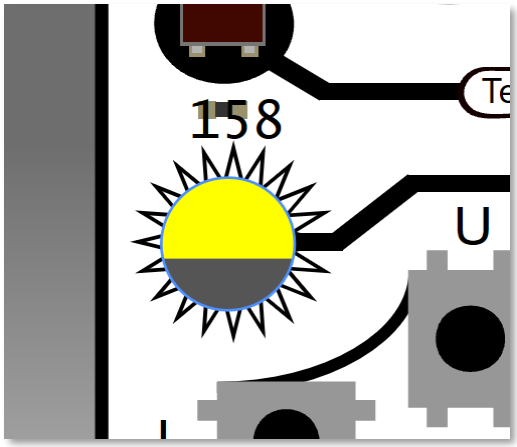
Що це означає? Якщо рівень освітлення менший, ніж 50, тоді треба написати «темно», а якщо ні, написати «світло» в одному рядку.

Не бійтеся відкривати «JavaScript»-секцію частіше і передивлятися як буде виглядати код.

```
forever(function () {
    if (input.lightLevel() < 50) {
        display.showString("Dark", 1)
    } else {
        display.showString("Light", 1)
    }
})
```

У симуляторі також є рівень освітлення. Щойно Light Sensor буде доданий у проект, симулятор почне відображати опцію для встановлення рівня освітлення.

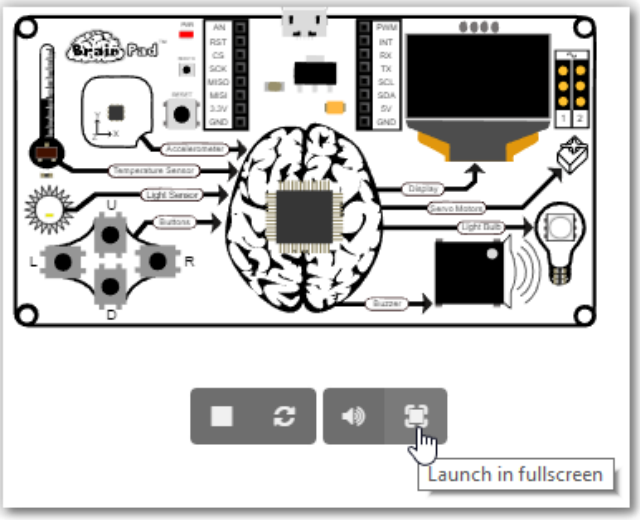
Light Sensor показує рівень освітлення, який можна змінювати, перетягати лінію вверх чи вниз. Це те ж саме, якщо би ви закрили датчик рукою на BrainPad. Коли ви змінюєте рівень освітлення вверх чи вниз, дисплей також відображає «темно» або «світло».



ПЕРЕВІРКА СЛУХУ

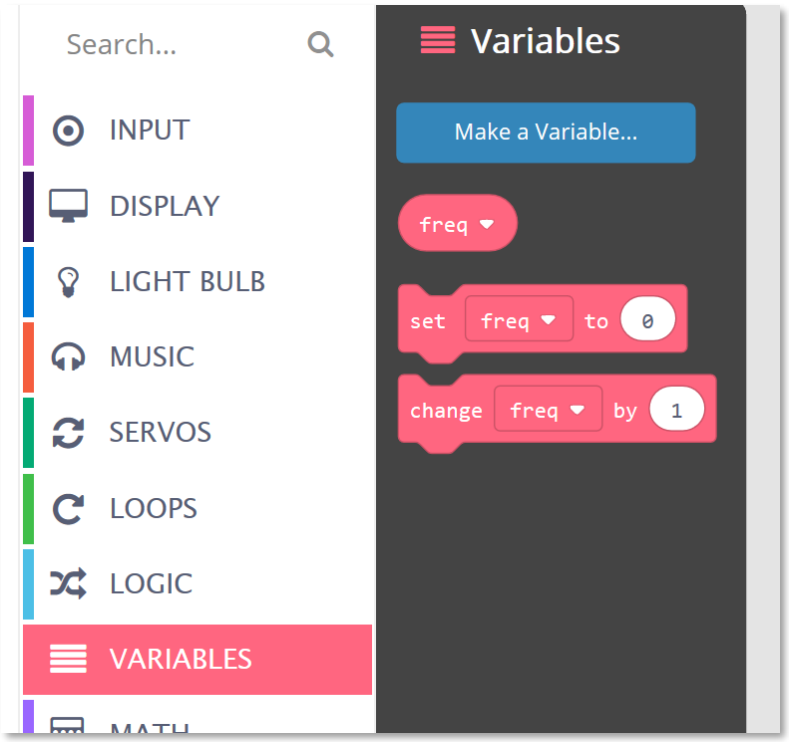
Людське вухо може чути частоти від низьких (приблизно від 20 Гц) до високих (біля 20,000 Гц). Собаки можуть чути ще вищі частоти – до 44,000 Гц. Ось чому ми не чуємо високочастотні свистки для собак, але тварини чують їх добре.

З віком ми втрачаємо здатність чути дуже високі частоти. Також втраті слуху сприяє високочастотний шум. Хочете перевірити, наскільки високі частоти ви здатні почути? Давайте розберемося!

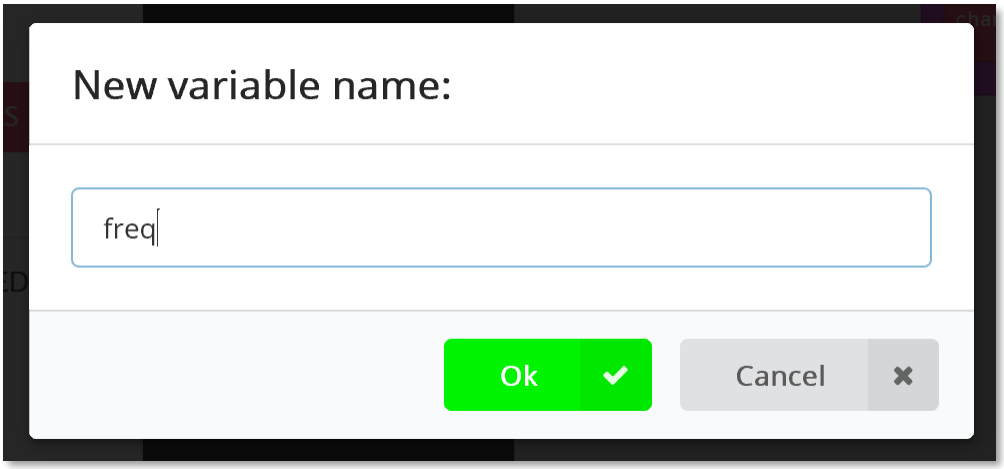


Нам потрібно написати програму, яка буде використовувати «змінні». Уявімо «змінні» у вигляді ящиків, у яких є свої імена. У такі ящики ми будемо складати які-небудь значення. У нашому випадку в «ящиках» будуть зберігатися частоти, які ми хочемо відтворювати у нашому динаміку.

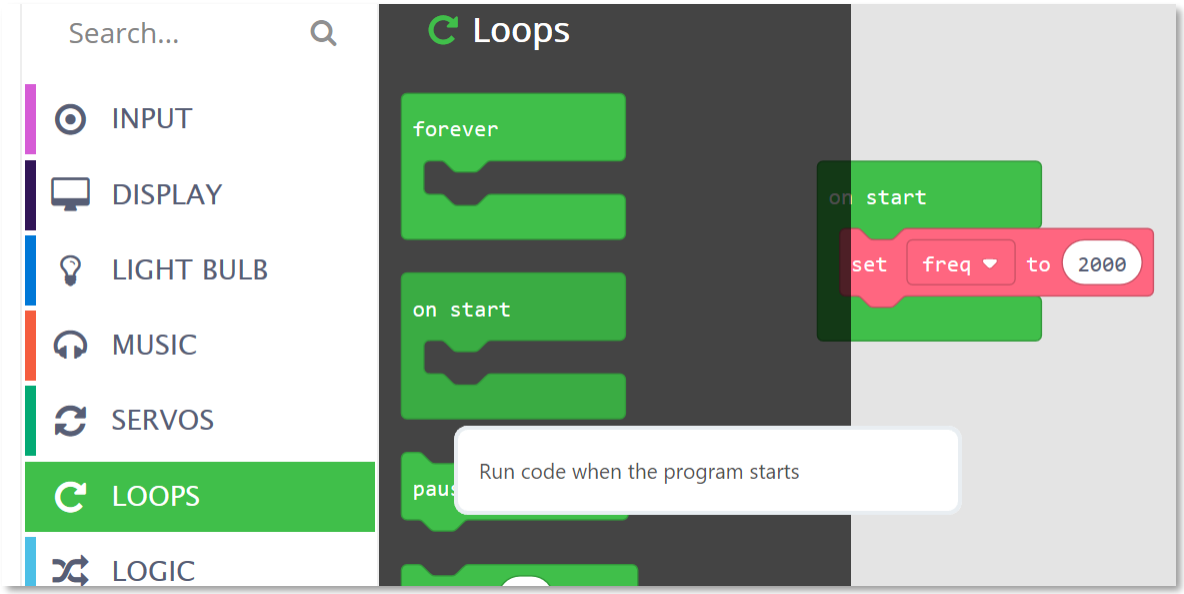
Перейдемо в меню VARIABLES й оберемо блок «Make a Variable. . .»



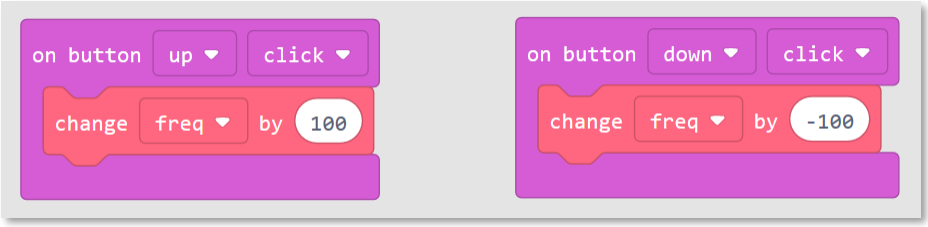
Дамо нашій змінній ім’я. Наприклад, «freq». Ця змінна буде зберігати частоту, яку ми будемо відтворювати у динаміку.



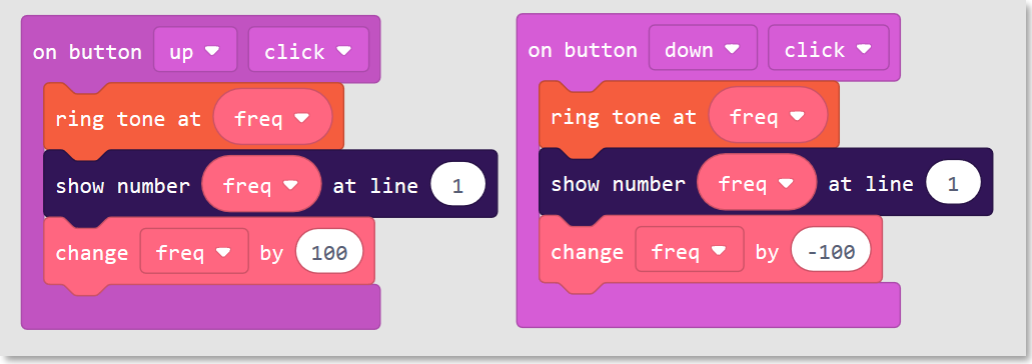
Тепер додаймо цикл «on start», який ви знайдете в меню LOOPS. Що б ви не додали у цей блок, це відбудеться, коли програма буде запущена. Установимо сюди змінну зі значенням 2000. Більшість людей може чути частоту 2000 Гц, отже, це хороший початок. Блок «set» знаходиться у меню VARIABLES.



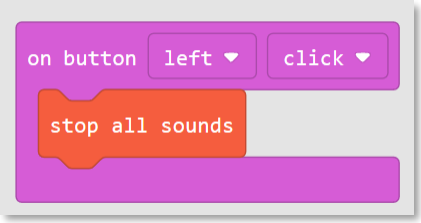
Зараз нам потрібно відтворити звук з цією частотою, але нам ще треба підвищити частоту, коли кнопка «вверх» натиснута і знизити, коли кнопка «вниз» натиснута. Нам потрібно змінити блок зі змінною: збільшити на 100, тобто, додати, коли натискаємо «вверх» і зменшити на 100, тобто, відняти, коли натискаємо кнопку «вниз».



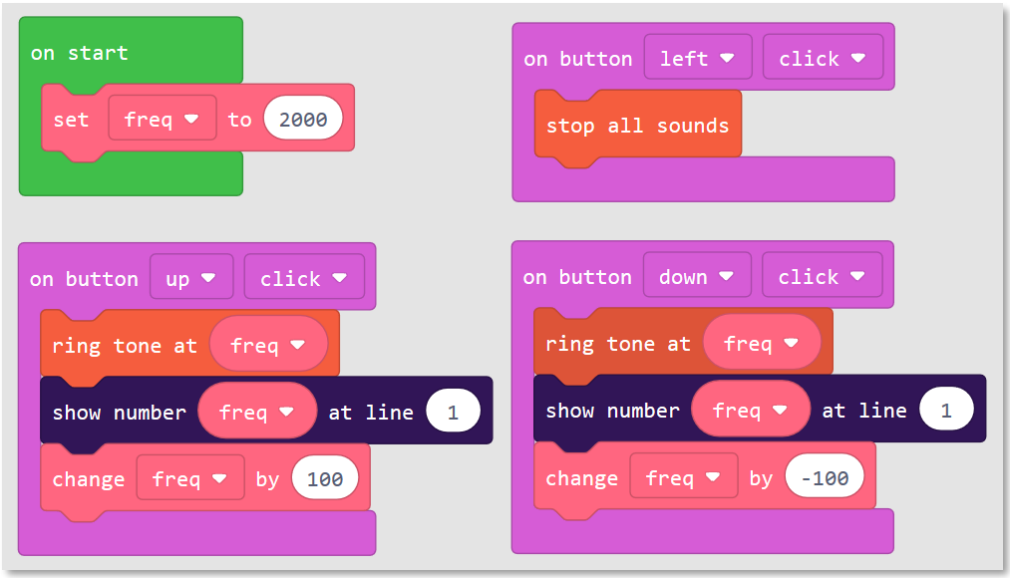
У нас є змінна, яку ми можемо змінювати так, як нам треба, і ми можемо генерувати звук (ring tone) і виводити частоту на дисплей (show number).



І останнє! Шум може сильно дратувати вас, тож давайте зупинимо звук натисканням кнопки «вліво».



Це вся програма.



СЕРВОДВИГУНИ

Серводвигун – це двигун, у якого є внутрішній електроланцюг, що дозволяє установлювати потрібне положення ротора електричними пульсами.

У серводвигуна є три контакти, які підключаються прямо до BrainPad. На жаль, у різних двигунів часто дроти різних кольорів. Щоб визначити як правильно підключити двигун, не звертайте увагу на центральний дріт – для нас важливі дроти, що знаходяться скраю. Світлий дріт повинен бути підключеним до контакту, над яким надрукований символ ~ на BrainPad. До речі, це сигнальний провід.

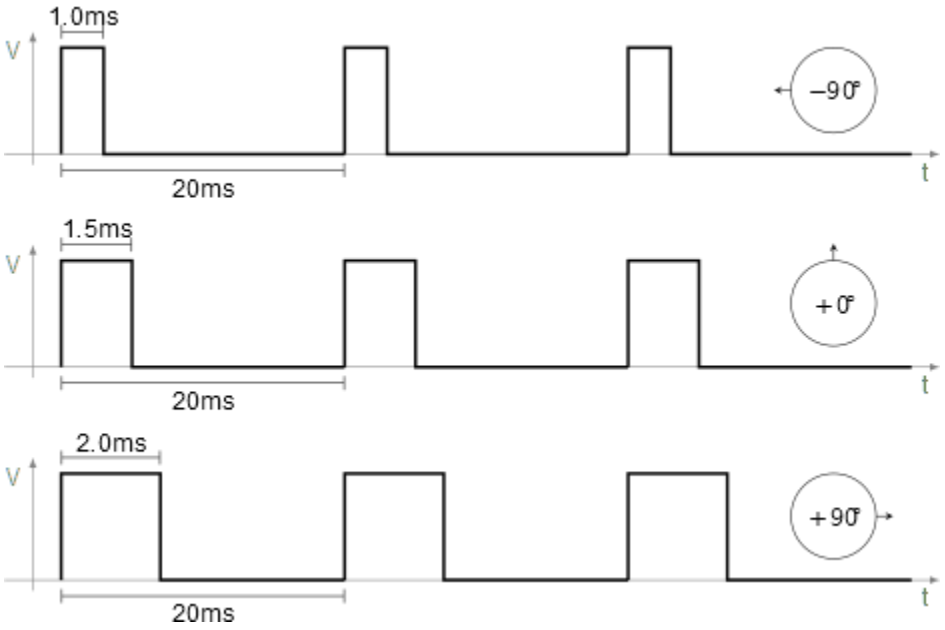
Серводвигуни часто використовують у радіокерованих моделях і їх можна знайти в інтернет-магазинах або в радіоаматорських магазинах неподалік від вас. Серводвигун живиться електрострумом від BrainPad, який бере електроенергію від вашого ПК або акумуляторної батареї (АКБ). Отже, ми рекомендуємо використовувати маленькі серводвигуни або мікросерво. Деякі серводвигуни споживають надто багато електроенергії для коректної роботи без додаткового джерела живлення.

Серводвигуни також бувають безперервними чи позиційними. Виглядають вони, як правило, однаково, але відрізняються функціями. У позиційних серводвигунах можна встановлювати ротор в певну позицію і утримувати його там, поки ви не зміните цю позицію. Натомість, безперервні обертаються в одному напрямку, поки ви не відправите сигнал зупинитися чи обертатися в іншому напрямку.

Якщо ви повернете ротор непідключеного двигуна рукою, то помітите, що він обертається лише на півоберта у кожному напрямку і зупиняється. Тоді як безперервний двигун можна обертати у будь-якому напрямку без обмежень. Так як у середні кожного серводвигуна є механізм, вам доведеться докласти певних зусиль, щоб повернути його.

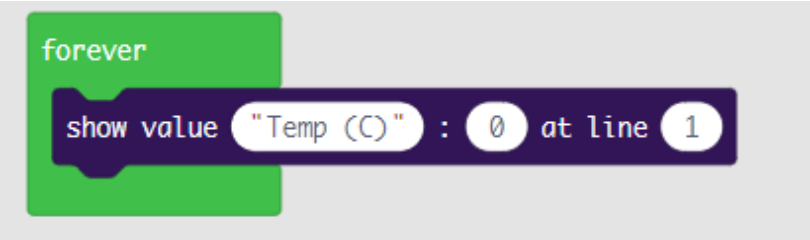
ПОЗИЦІЙНІ СЕРВОДВИГУНИ

Як було згадано вище, позиційні двигуни можна обертати в певну позицію (чи кут), після чого він в цій позиції і залишиться, поки ви не зміните цей кут. Якщо ви спробуєте повернути серводвигун рукою і в певній позиції він зупиниться, значить це позиційний серводвигун. Пульсуючий сигнал, який надсилає BrainPad, каже двигуну, на який кут чи позицію йому обернутися.

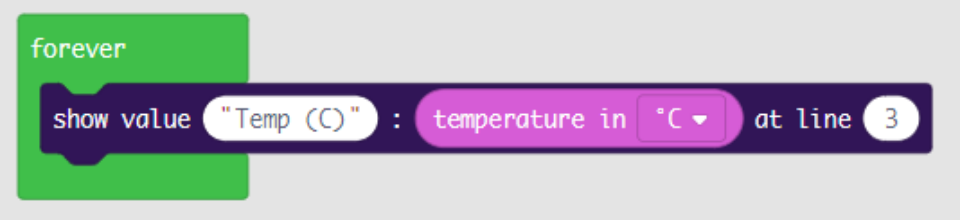


Існує багато способів використовувати позиційні серводвигуни. Наприклад, ми сконструювали великий термометр з циферблатом. Код, який ми для цього використали, дуже простий і для нього навіть не потрібні змінні.

У вічному циклі додайте блок «show value» (його ви знайдете в меню у секції «дисплей»). У першому овалі наберіть «Temp (C)» у дужках, як показано нижче.

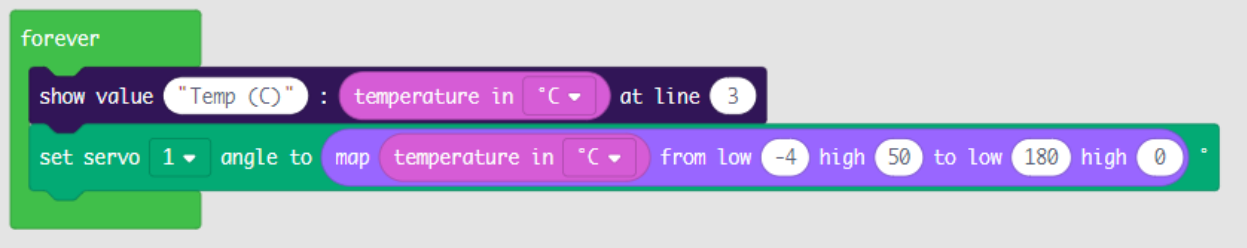


З категорії «input» оберіть блок «temperature in °C» і перетягніть його у другий овал. У третьому овалі змініть значення і покладіть 3. Тепер значення температури буде відображатися на дисплеї BrainPad у третьому рядку. Ваші блоки повинні виглядати так:



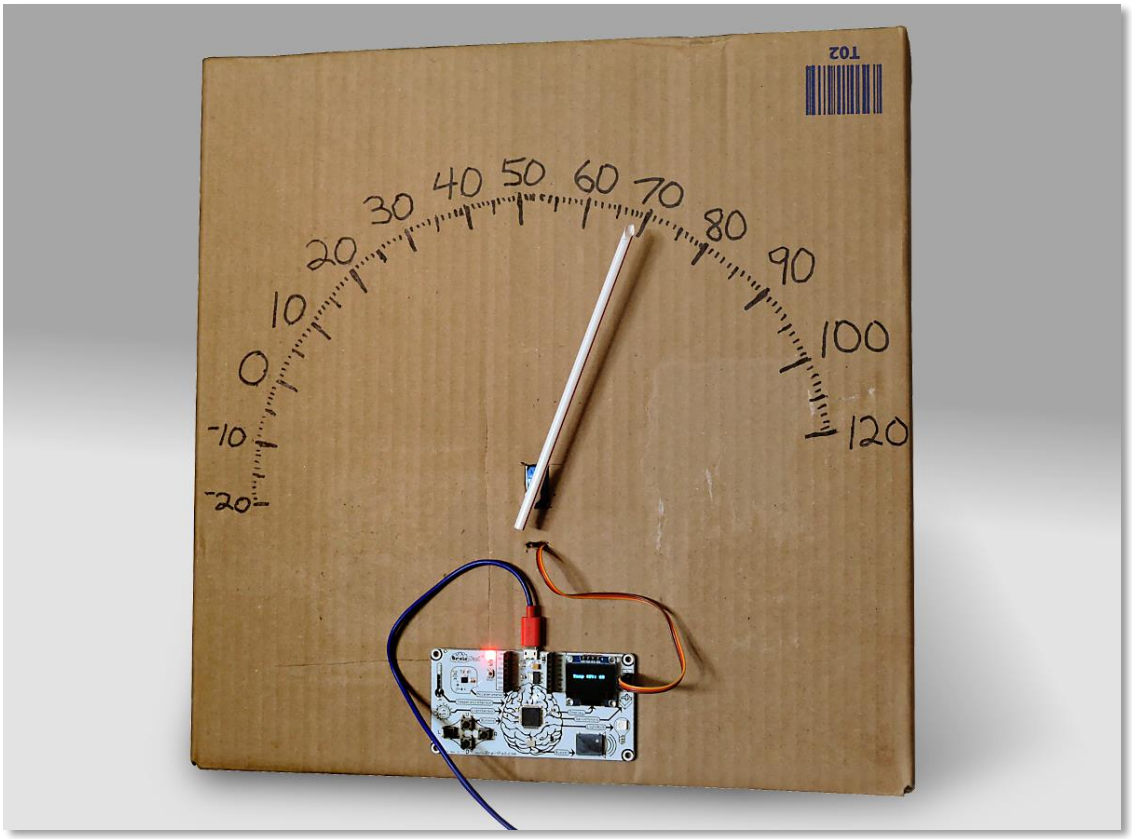
Досить просто ми зробили термометр, який показує температуру на дисплеї BrainPad. Тепер нам треба відобразити температуру в кут і надіслати це значення серводвигуну, щоб повернути його на циферблаті нашого термометра. Налаштуємо термометр так, щоб він вимірював температуру у межах від -4° до 50° за Цельсієм.

Нам потрібно відобразити температуру (від -4 до 50) в кут (від 0 до 180). Якщо ви не надто дружите з математикою, не хвилюйтесь! У MakeCode є блок "map", який вирішить усі математичні задачі за вас. Знайти його можна в меню "MATH". Ми можемо закінчити нашу програму, додавши усього один рядок:



Можливо, ви звернете увагу на те, що низькій температурі відповідає 180 градусів, а високій 0 градусів у нашому "map" блоці. Все навпаки? Причина в тому, що кут в 0 градусів відповідає повороту серводвигуна вправо, а 180 градусів – уліво. Якби ми це не врахували, то наш термометр міряв би температуру навпаки.

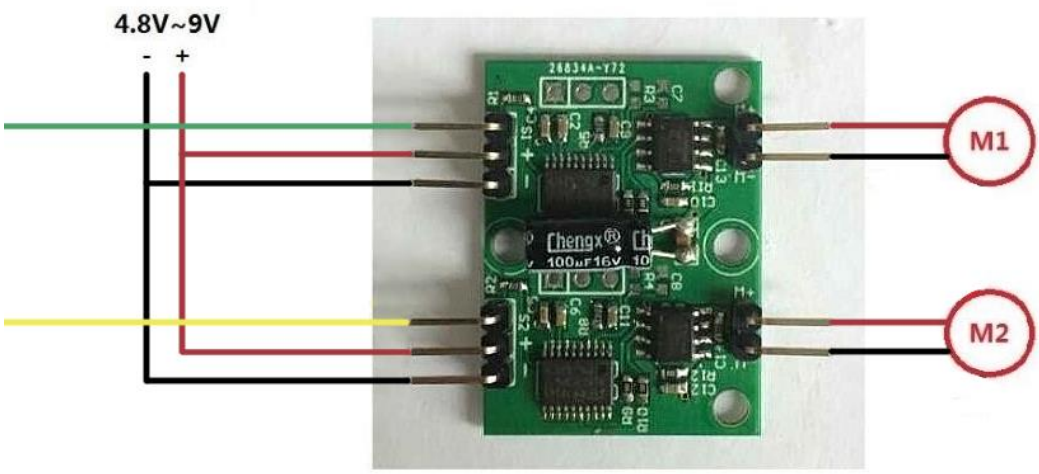
Ось закінчений термометр, зроблений з картонної коробки і соломинки, приклеєної до серводвигуна (у США температуру вимірюють у градусах по Фаренгейту. Шкала температури на термометрі, який на картинці відрізняється від того, що повинно вийти у вас. Вам потрібно відобразити градуси за Цельсієм, де замість -20 буде -4, а замість 120 буде 50 градусів).



БЕЗПЕРЕРВНІ СЕРВОДВИГУНИ

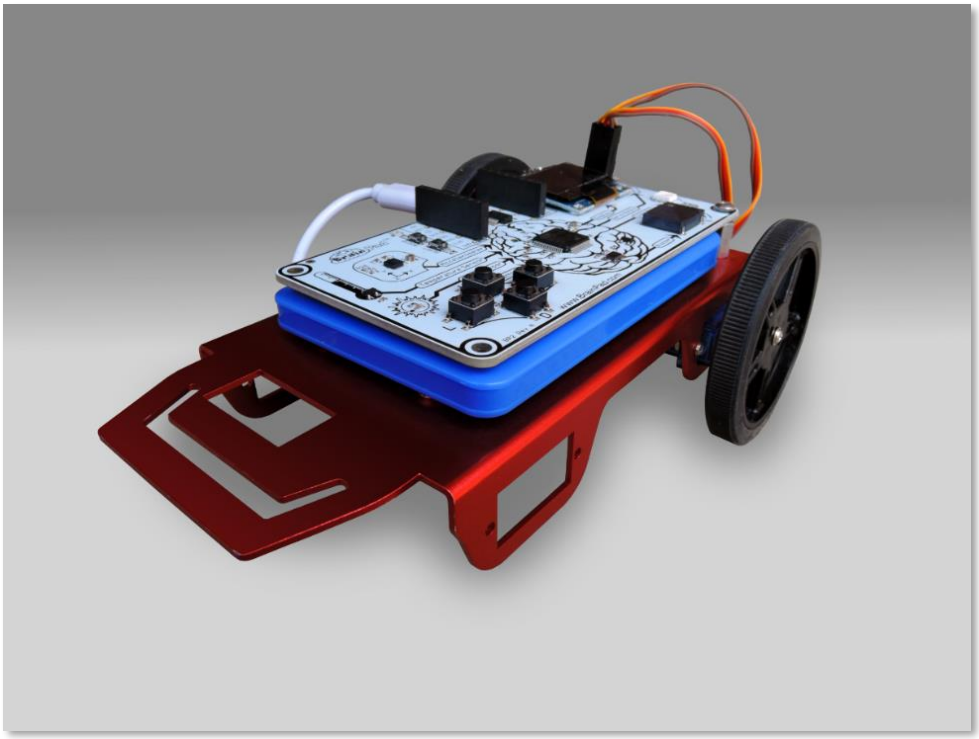
Другий тип серводвигунів – безперервні, які, як видно із назви, можуть обертатися безперервно. У даному випадку, пульси встановлюють швидкість і напрямок руху двигуна. До BrainPad можна підключити два двигуна. Керуючи напрямком і швидкістю, ви легко можете сконструювати робота. В інтернет-магазинах можна швидко і недорого купити все, що нам потрібно.

Цей набір можна легко встановити на картонну коробку, але пізніше ми знайшли ось такі шасі від компанії FEETECH.



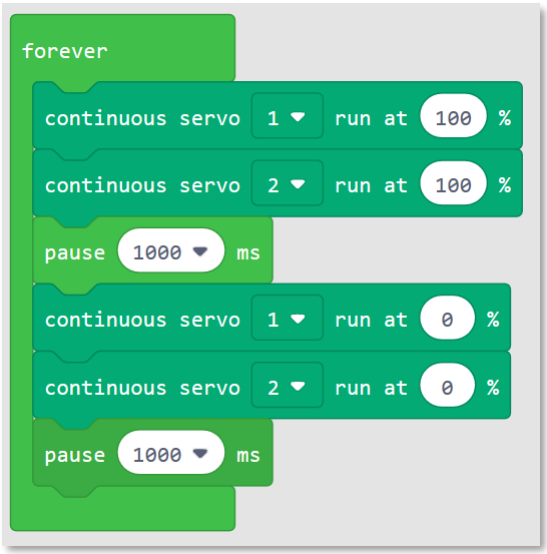
Тут є невеликі труднощі. У комплекті цього робота є колеса і двигуни. Двигуни виглядають як серво, але під'єднуються вони як звичайні двигуни. Ми зрозуміли це з того, що у них лише по два проводи, а не по три, як у наших серводвигунів. В наборі також була електронна плата, яка, як пізніше стало зрозуміло, перетворює прості двигуни в серво.

Можна працювати і з цим, але для занять у класі це досить складно. Тому ми відклали цей перетворювач та двигуни і вирішили встановити на нашого робота безперервні серводвигуни, про які ми говорили вище. Наш BrainPad чудово підходить до цього шасі, ніби вони були створені один

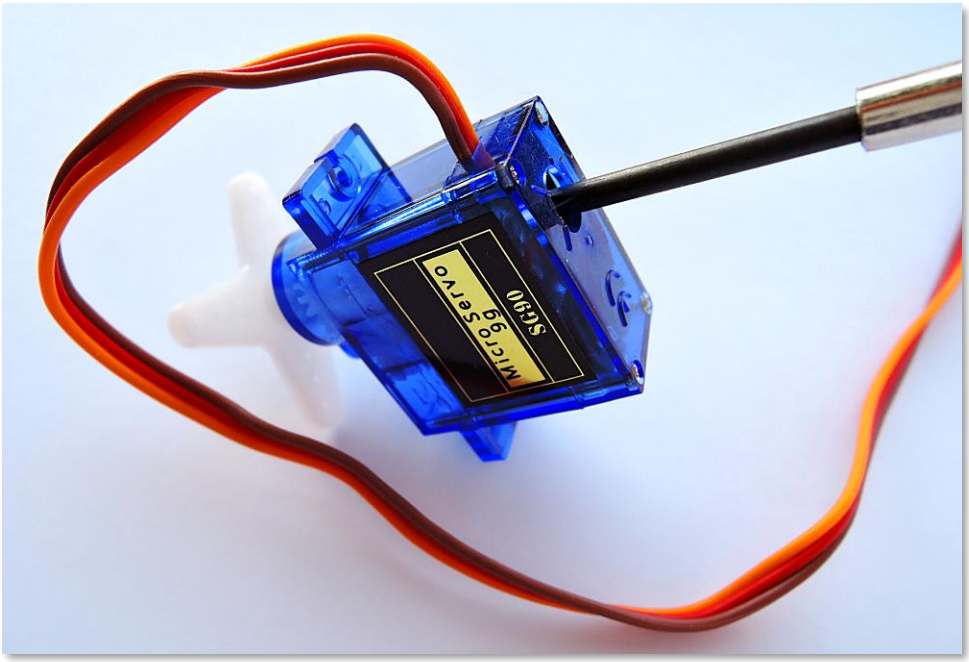


для одного. Також для цього проекту я підібрав зовнішню АКБ для мобільного телефону (power bank), яка також підійшла за розмірами до BrainPad.

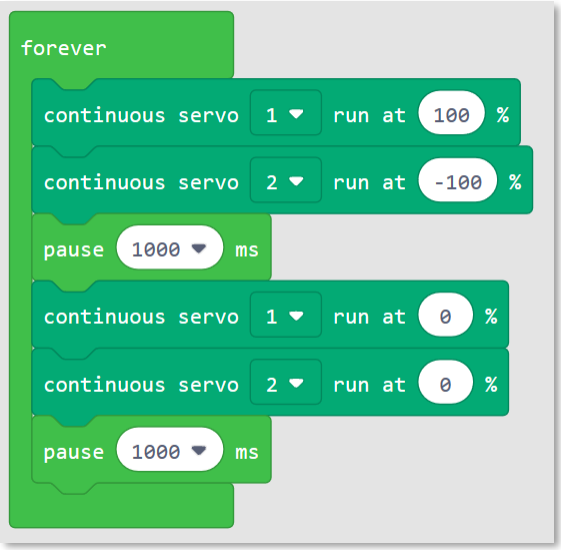
Нарешті цей робот готовий до запуску! Щоби він рухався вперед, обидва двигуни повинні обертатися вперед. Оскільки один з двигунів розвернутий «обличчям» справа, в другий зліва, то один з них повинен рухатися в протилежному напрямку. На словах це звучить заплутано, але щойно ви спробуєте все зібрати і запустити, вам усе стане зрозуміло. Цей код рухає обидва двигуна вперед на повній швидкості одну секунду, зупиняє їх на одну секунду і потім знову повторює цей процес.



У випадку, якщо двигуни продовжують обертатися при виставленій швидкості 0, то вони потребують калібровки. Більшість серводвигунів можна налаштувати. Вам знадобиться для цього маленька викрутка.



Повернімося до нашого робота. Ви звернули увагу на те, що один з двигунів рухається в неправильному напрямку? Змініть напрямок його рухів на протилежний.



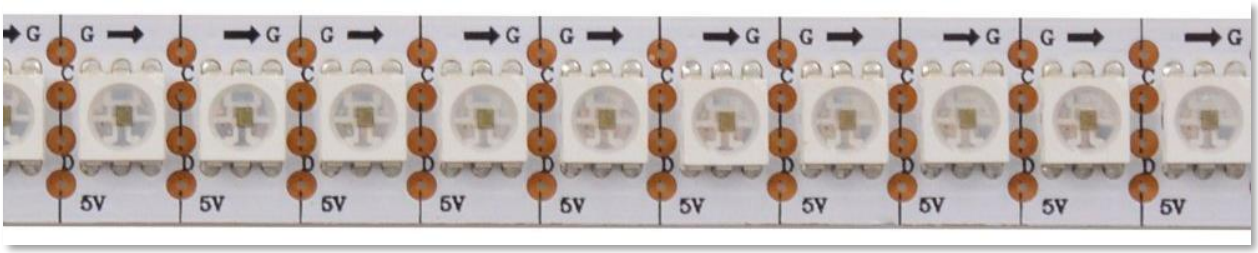
Цей код – частина нашого проекту «Sun Seeker» («Шукач Сонця»). Ви можете знайти його тут https://docs.brainpad.com/projects/sun-seeker.html. Обов'язково перегляньте відео. Робот використовує датчик світла, щоб визначити – знаходиться він під сонцем чи у тіні. Якщо в тіні, то серводвигуни будуть його розвертати, поки він не знайде сонце.

РІЗДВЯНІ ВОГНІ

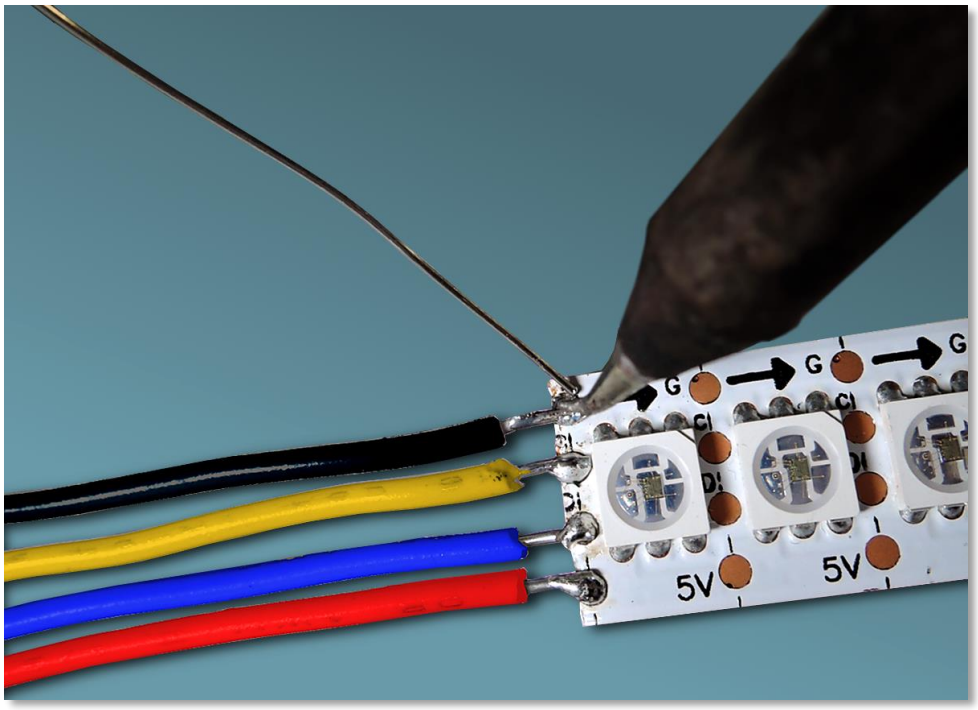
Ви можете створити багато проектів, використовуючи вбудовані у BrainPad опції, але невдовзі ви захочете підключити до нього зовнішні пристрої. Так як це виходить за межі нашої книги, ми підготували для вас коротку інструкцію, щоби показати, які проекти можна реалізовувати.

У цьому проекті ми будемо використовувати стрічку світлодіодів, до кожного з яких можна повертатися за певною адресою. Тобто, будь-яким світлодіодом можна керувати окремо. Кожен світлодіод можна запалити певним кольором та яскравістю, використовуючи лише два дроти. Цими дротами світлодіоди отримують від нас дані, зашифровані пульсуючими сигналами.

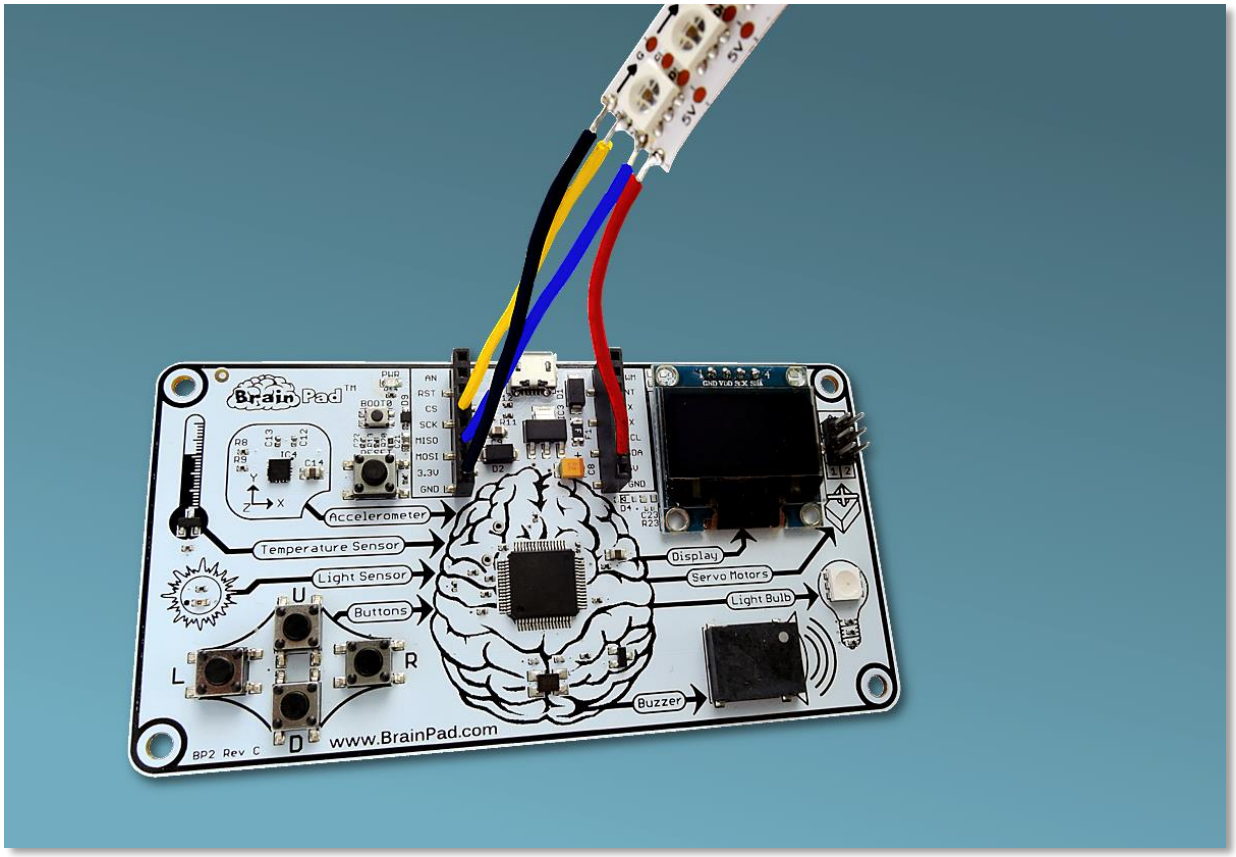
Вам необхідно знайти світлодіодну стрічку з контролером APA102. Цей контролер використовує SPI протокол, який доступний на BrainPad. Ці стрічки підключаються до джерел живлення з різним вольтажем. Нам потрібні ті, які працюють від 5В, адже у нас є доступ до 5В.



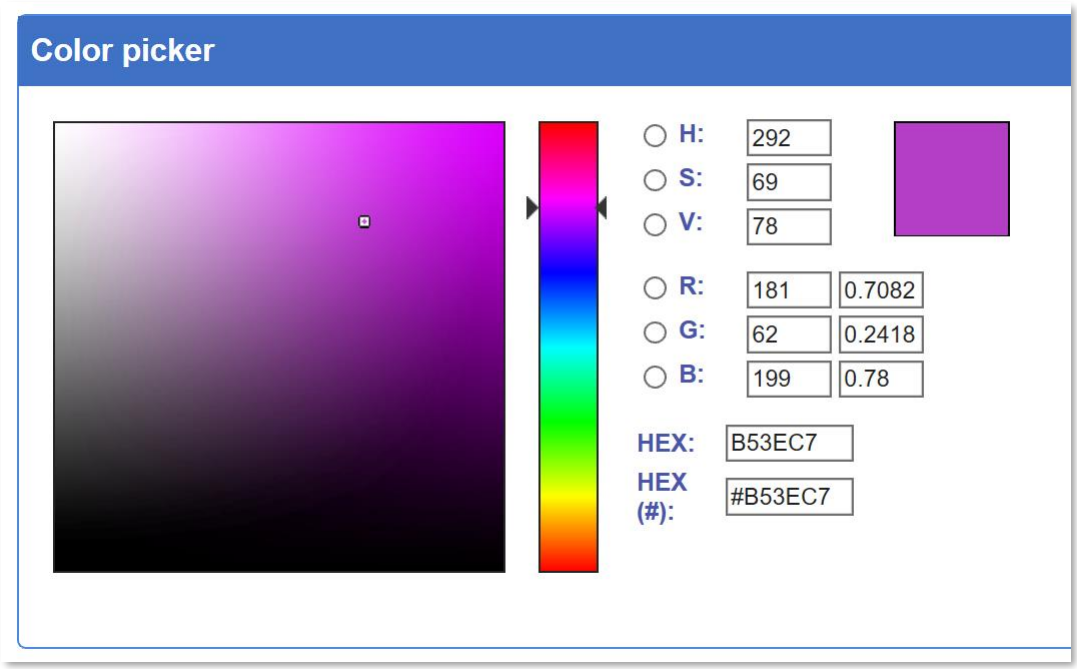
Цю стрічку можна розрізати у будь-якому місці. Щоб вона не споживала надто багато енергії, залишимо 25 світлодіодів. Тепер треба припаяти кілька дротів. Стрілка на стрічці вказує до якої сторони треба припаювати дроти. І так чотири дроти: G-Ground (земля), C-Clock (тактуючий сигнал), D-Data (дані) и 5V (5В).


G
G
G
5V
5V

GND і 5V зрозуміло позначені на додаткових коннекторах BrainPad'у. C-Clock повинен бути підключеним до SCK на BrainPad. D-Data треба підключити до MOSI BrainPad.



У кожному світлодіоді є три маленькі світлодіодики – червоний, синій і зелений. Керуючи інтенсивністю підсвічування кожного з них можна створити колір, який вам подобається. Є кілька онлайн-інструментів, які допоможуть вам обрати потрібний колір. Хороший приклад ви можете знайти тут http://www.rgbtool.com. Вам потрібно встановити значення R (червоний), G (зелений), і B (синій).



Команди, які ми надсилаємо світлодіодам, використовують шину SPI, яка доступна в BrainPad. Щоб почати послідовно надсилати команди світлодіодам, потрібно відправити блок стартового коду. В даному випадку це просто чотири нулі.

Після стартового блоку кожному світлодіоду необхідно чотири числа, щоби ним керувати. Перше число керує загальною яскравістю всіх світлодіодів на стрічці. Давайте встановимо перше число, наприклад, 255, воно виставить повну яскравість.

Після першого числа вводимо три інших у діапазоні від 0 до255. Так ми керуємо яскравістю синього, зеленого і червоного у кожному світлодіоді. Щоби запалити світлодіод синім кольором, вам потрібно 255,0,0. Щоб відключити його, вам потрібно відправити 0,0,0. Щоб світлодіод горів яскравим білим світлом, потрібно відправити 255,255,255.

Світлодіоди у стрічці підключені послідовно один за одним. Перший колір відправляється першому світлодіоду, другий – другому тощо. Світлодіоди будуть безперервно світитися, поки не отримають новий стартовий блок (чотири нулі), після цього послідовно повторюється процес.

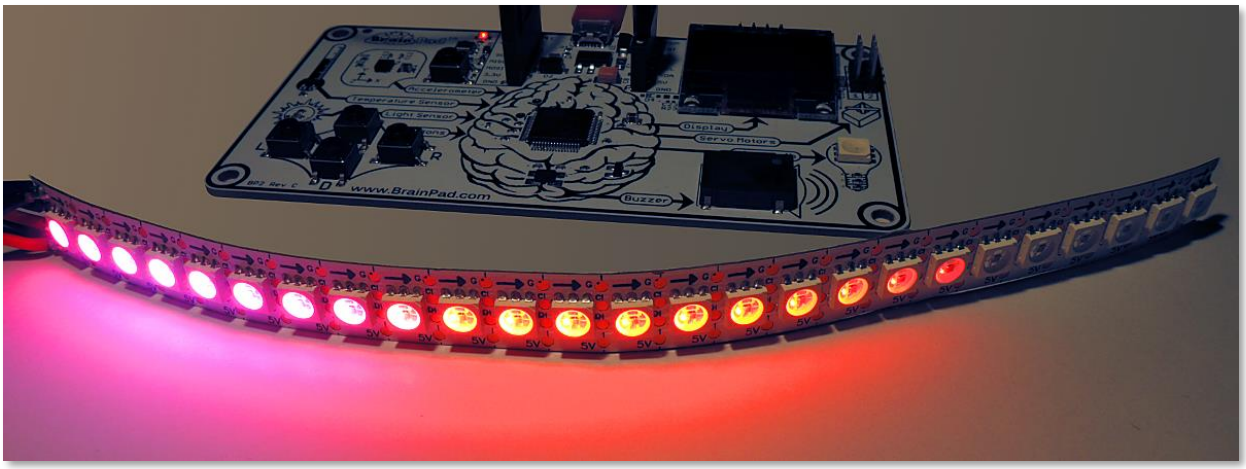
У цьому прикладі досить багато блоків, тому легшим шляхом буде переключитися на JavaScript, скопіювати і вставити цей код JavaScript у редактор в MakeCode:

```
let center = 12
let startRed = 0
let startGreen = 0
let startBlue = 0
```

```
let red = 0
let green = 0
let blue = 0
let data: Buffer = pins.createBuffer(4)
let dummy: Buffer = pins.createBuffer(4)

pins.spiFrequency(1000000)

function SendStart() {
    data[0] = 0
    data[1] = 0
    data[2] = 0
    data[3] = 0

  pins.spiTransfer(data, dummy)
}

function SendRGB(red: number, green: number, blue: number){
    data[0] = 255
    data[1] = blue
    data[2] = green
    data[3] = red

  pins.spiTransfer(data, dummy)
}

forever(function (){
    startRed = Math.constrain(startRed + Math.randomRange(-20, 20), 0, 255)
    startGreen = Math.constrain(startGreen + Math.randomRange(-20, 20), 0, 255)
    startBlue = Math.constrain(startBlue + Math.randomRange(-20, 20), 0, 255)

    SendStart()
```

```
  for (let index = 0; index <= 24; index = index + 1) {
        red = Math.constrain(startRed - Math.abs(index - center) * 16, 0, 255)
        green = Math.constrain(startGreen - Math.abs(index - center) * 16, 0, 255)
        blue = Math.constrain(startBlue - Math.abs(index - center) * 16, 0, 255)

        SendRGB(red, green, blue)
  }
})
```

Якщо ви зробили все правильно, то побачите різнокольорові вогники. Спробуйте змінити код і подивитися як це вплине на результат.



ВИСНОВКИ

Microsoft MakeCode і BrainPad – це чудові інструменти для вивчення програмування, які відмінно доповнюють одне одного. Завдяки симулятору ви можете тестувати код навіть якщо у вас ще немає BrainPad. У майбутньому ви можете завантажити програму на BrainPad, просто скопіювавши завантажений файл у вікно BrainPad.

У наступних розділах ми вийдемо за межі і будемо програмувати на C# або Visual Basic в Microsoft Visual Studio. «Виходимо за межі» - це додатковий, але дуже потрібний крок, до якого рекомендуємо перейти, коли ви почнете впевнено почувати себе у MakeCode. У більшості випадків MakeCode – це все, що вам потрібно.

GO BEYOND – ВИХОДИМО ЗА МЕЖІ

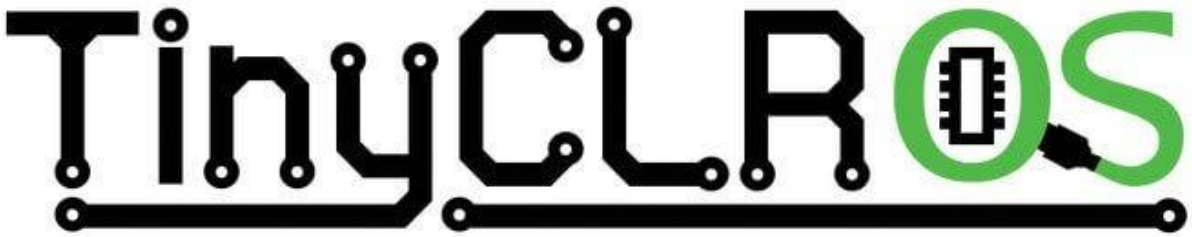
Ви тут, тому що хочете навчитися програмувати як професіонали. Ми будемо користуватися Microsoft Visual Studio, щоб програмувати на C# або Visual Basic. У вас будуть повні можливості налагодження, а також багатофункціональне середовище розробки.

Попередньо рекомендуємо ознайомитися з Microsoft MakeCode.

Отож, пристебніть паски безпеки, і ми вирушаємо в серйозну подорож країною професійного програмування, яка може перетворитися у майбутню кар'єру!

TINYCLR OS™

Щоби BrainPad працював з .NET на C # и Visual Basic, вам знадобиться TinyCLR OS. Ця мініатюрна операційна система створює всю магію. Вона інтерпретує скомпільований код .NET і містить тисячі таких сервісів, як потоки, керування пам'яттю і налагодження. Це також .NET, який використовується для розробки програмного забезпечення для телефонів та комп'ютерів з тією різницею, що TinyCLR OS працює на мінікомп'ютерах, таких як BrainPad. Тож TinyCLR OS – це частина стандарту .NET.



VISUAL STUDIO

Теоретично будь-який компілятор .NET може використовуватись з TinyCLR OS; однак TinyCLR підтримується тільки Microsoft Visual Studio. Visual Studio версія Comunity представляє собою повнофункціональне середовище програмування, яке вважається одним з найкращих (якщо не кращим), доступним, а найголовніше – безкоштовним! Для установлення Visual Studio вам знадобиться сучасний ПК з Windows. Рекомендується використовувати Windows версії 10.

Зверніть увагу, що ви можете використовувати ту ж Microsoft Visual Studio і ті ж навички, які ви вивчаєте, програмуючи BrainPad, щоби почати писати додатки для багатьох інших пристроїв, таких як смартфони, ПК і навіть ігрові консолі Microsoft Xbox!

СИСТЕМНІ НАЛАШТУВАННЯ

Microsoft Visual Studio версія Community безкоштовна, але це не іграшка, і вона не маленька. Вам знадобиться певний час для налаштування системи. Повні інструкції викладені на веб-сайті документації BrainPad за адресою https://docs.brainpad.com/go-beyond/system-setup.html.

У результаті вам треба буде завантажити і встановити:

1. Microsoft Visual Studio, версія Community. Не забудьте обрати опцію «.NET desktop development» під час налаштування.

2. GHI Electronics TinyCLR OS project system

Якщо ви заплутались, то завжди можете зайти на наш форум і отримати відповіді на всі питання https://forums.ghielectronics.com/c/brainpad

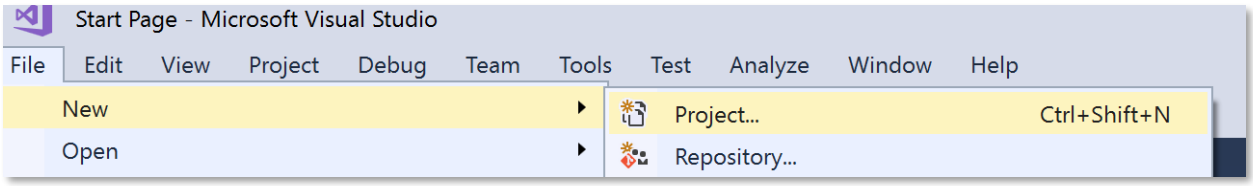
Тепер, коли комп'ютер налаштований, вам потрібно завантажити TinyCLR OS на BrainPad. Завантажте файл прошивки TinyCLR OS тут https://docs.brainpad.com/resources/downloads.html, а потім завантажте його на BrainPad так же, як робили це з Microsoft MakeCode:

1. Утримуйте кнопку «reset» протягом приблизно трьох секунд. Лампочка стане зеленою.
2. ПК виявить запам'ятовуючий пристрій і відкриє нове вікно.
3. Збережіть прошивку TinyCLR OS на BrainPad. Також ви можете перетягнути або скопіювати файл і вставити його у вікно BrainPad.

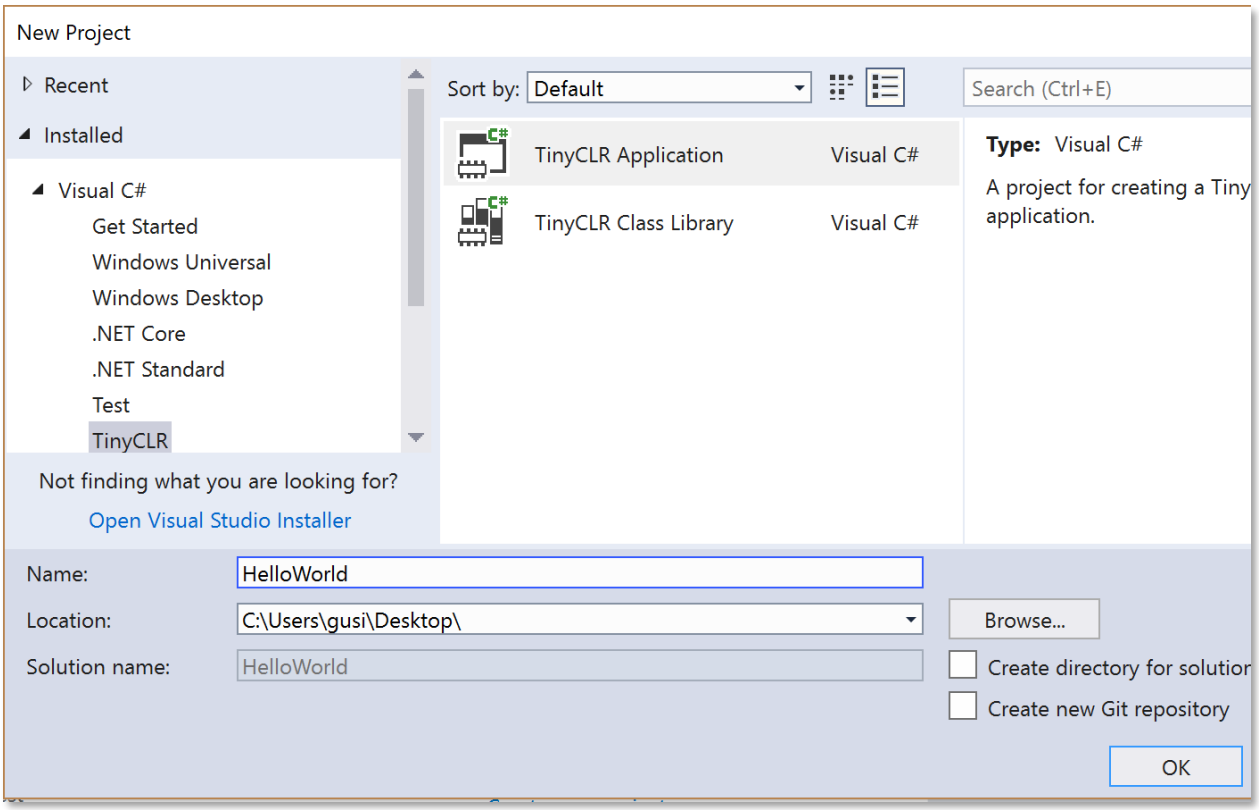
HELLO WORLD

Зазвичай, коли програмують новий пристрій чи починають використовувати нові інструменти для програмування, щоби впевнитись, що все працює правильно, пишуть дуже просту програму. Ця програма зазвичай називається «Привіт, світ» або «Hello world». Давайте зробимо це зараз.

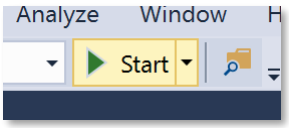
Відкрийте Microsoft Visual Studio і почніть новий проект. У меню «File» оберіть «New» і «Project» (File > New > Project).



У Visual C # повинна бути опція TinyCLR. Якщо немає, то ви не встановили TinyCLR OS project system. Цей проект ми назвали «HelloWorld» і розмістили його на своєму робочому столі.



Тепер ви можете запустити цю пусту програму на своєму BrainPad. Вона не зробить нічого корисного, але система перевірить, чи все гаразд і чи все працює так, як ми очікуємо. Підключіть BrainPad і натисніть «Start».

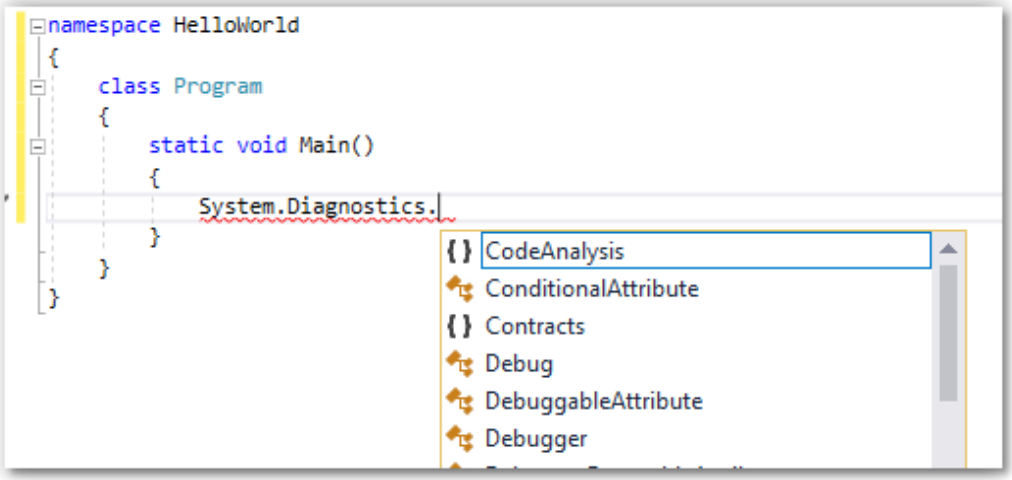


Спостерігайте за процесами, які відображаються у вікні виведення. Якщо цього вікна не видно, натисніть Alt + Ctrl + O на клавіатурі чи оберіть «Output» в меню «View» (View > Output).

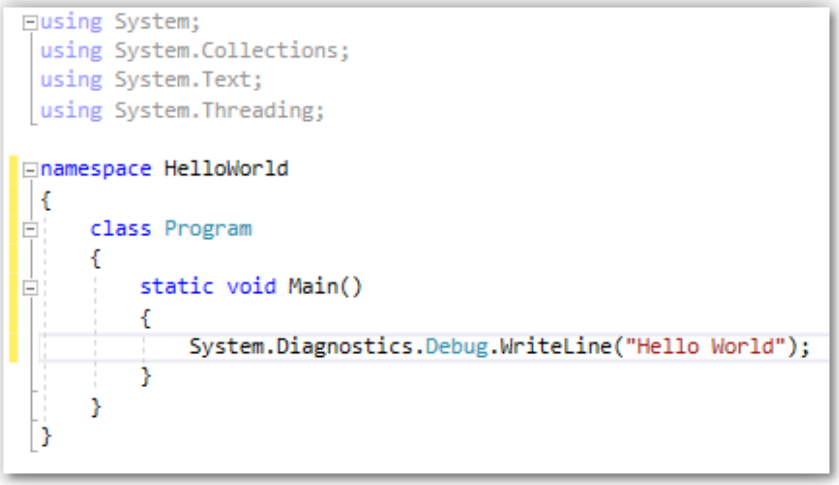
Тепер ми готові сказати BrainPad'у «Hello World». Спочатку ми виведемо на друк «Hello World» у вікні виведення Visual Studio. Пізніше виведемо його на дисплеї BrainPad. Додайте цей рядок в програму прямо під лівою дужкою ({), яка йде наступною за Main ().

```
System.Diagnostics.Debug.WriteLine("Hello World");
```

Зверніть увагу, що Visual Studio автоматично «здогадується», що ви хочете набрати на клавіатурі.



Коли все буде виконано, код повинен виглядати ось так:



Натисніть на «Start» і спостерігайте за вікном виведення.

```
The debugging target runtime is loading the application assemblies and starti
Ready.

Done.

Waiting for debug commands...

'GHIElectronics.TinyCLR.VisualStudio.dll' (Managed): Loaded 'C:\Users\gusi\De
The thread '<No Name>' (0x2) has exited with code 0 (0x0).
Hello World
The thread '<No Name>' (0x1) has exited with code 0 (0x0).
The program '[3] TinyCLR application: Managed' has exited with code 0 (0x0).
```

Ви знайшли «Hello World»? Це був хороший початок, але не надто захопливий. Спробуймо вивести на екран кілька рядків. Замініть раніше доданий рядок цими чотирма.

```
System.Diagnostics.Debug.WriteLine("The");
System.Diagnostics.Debug.WriteLine("BrainPad");
System.Diagnostics.Debug.WriteLine("is");
System.Diagnostics.Debug.WriteLine("amazing!");
```

Помістіть курсор у перший рядок і натисніть функціональну клавішу F9. Це додасть точку інспекції коду або простіше «breakpoint». Ви можете зробити те саме з меню «Debug».

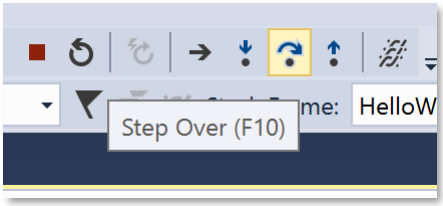
```
namespace HelloWorld
{
    class Program
    {
        static void Main()
        {
            System.Diagnostics.Debug.WriteLine("The");
            System.Diagnostics.Debug.WriteLine("BrainPad");
            System.Diagnostics.Debug.WriteLine("is");
            System.Diagnostics.Debug.WriteLine("amazing!");
        }
    }
}
```

Запустіть програму, як ви робили це раніше. Як тільки виконання коду дійде до breakpoint,

```
 6  namespace HelloWorld
 7  {
 8      class Program
 9      {
10          static void Main()
11          {
12              System.Diagnostics.Debug.WriteLine("The");
13              System.Diagnostics.Debug.WriteLine("BrainPad");
14              System.Diagnostics.Debug.WriteLine("is");
15              System.Diagnostics.Debug.WriteLine("amazing!");
16          }
17      }
18  }
19
```

програма зупиниться у тому ж рядку.

Програма зупиниться прямо перед тим, як цей рядок повинен бути виконаним. Натисніть клавішу F10 або натисніть кнопку покрокової інспекції коду.

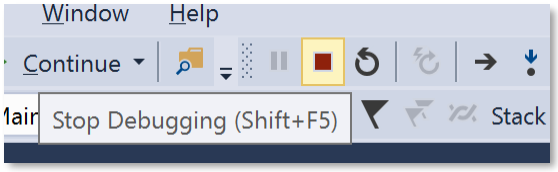


Це змусить програму виконати один рядок коду за один крок. Натисніть F10 і спостерігайте за вікном виведення. Це називається покроковою інспекцією коду. Ці дії дуже корисні тоді, коли ми налагоджуємо нашу програму, і дають нам можливість зрозуміти, чому щось не працює належним чином.

НЕСКІНЧЕННІ ПРОГРАМИ

На телефоні чи ПК є сенс закривати чи закінчувати програми. Однак, на невеликих пристроях, які працюють під керуванням мікроконтролерів, програми зазвичай не закінчуються. Наприклад, мікрохвильова піч. У неї є невеликий «мозок» всередині, який завжди контролює клавіші для запуску процесів і запускає годинник на екрані. Якщо ви не відключаєте мікрохвильову піч, ця програма завжди працює. Це називається безкінечним циклом і часто використовується у програмуванні.

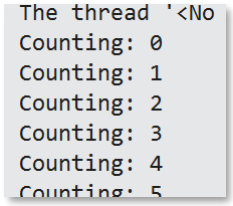
Давайте створимо програму-лічильник на BrainPad, яка запускається і працює завжди. Тобто, BrainPad буде рахувати один раз на секунду і ніколи не зупинятися. Зверніть увагу, що перед тим, як ви хочете щось змінити, вам необхідно зупинити будь-яку запущену програму. Ви можете зробити це, натиснувши кнопку «Stop». Нижче наведений код, який повинен виглядати ось так.



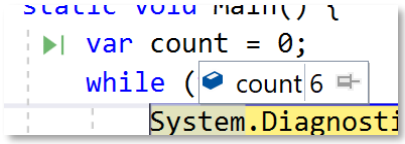

```
namespace HelloWorld {
    class Program {
        static void Main()
        {
            var count = 0;

            while (true)
            {
                System.Diagnostics.Debug.WriteLine("Counting: " + count);
                Thread.Sleep(1000); // Wait one second

                count = count + 1;
```

```
            }
        }
    }
}
```

Вікно виведення покаже лічильник.

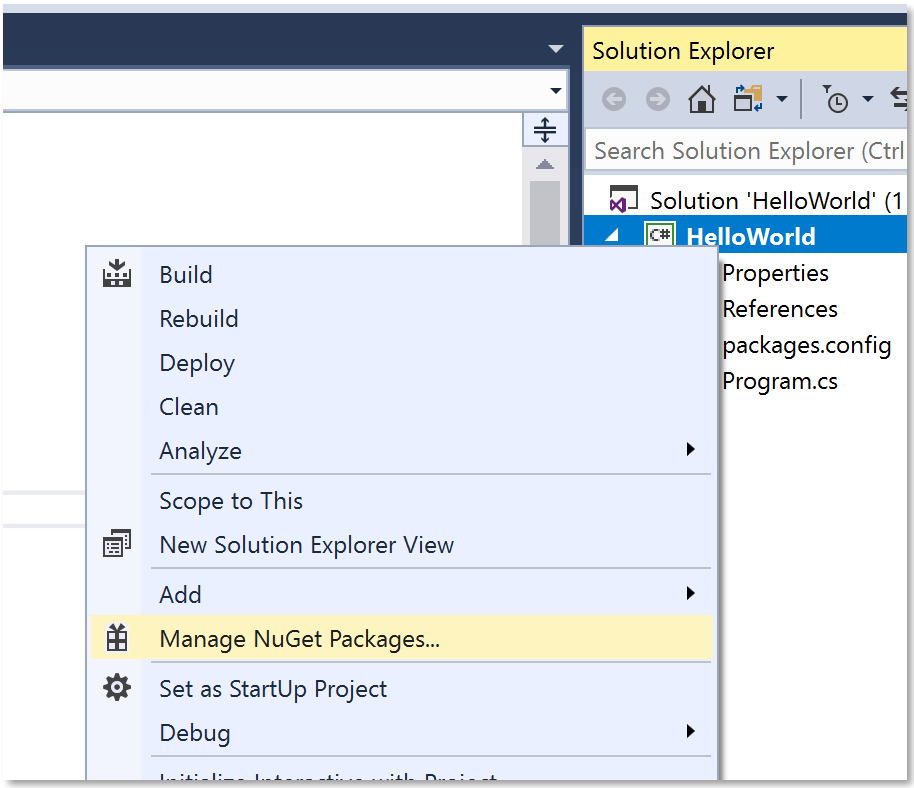


Коли програма запущена, вставте «breakpoint» в рядок лічильника. Звідти ви можете «пройтись» по коду, як ми робили це раніше. Тепер наведіть курсор миші на нашу змінну «count»" і Visual Studio покаже вам дане значення змінної. Коли ми зупинили програму, вона дорівнювала 6. До речі, перевірка змінних – ще одна дуже корисна функція для налагодження програм.

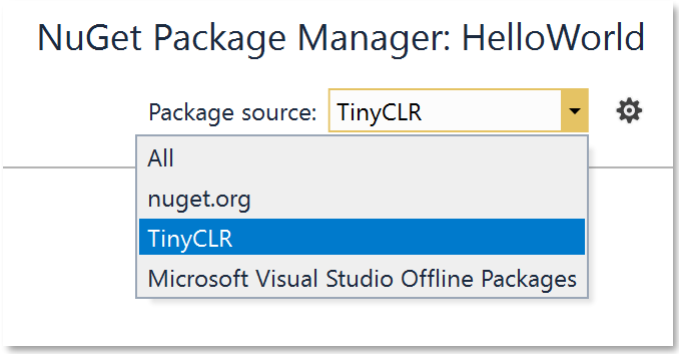


BRAINPAD БІБЛІОТЕКИ

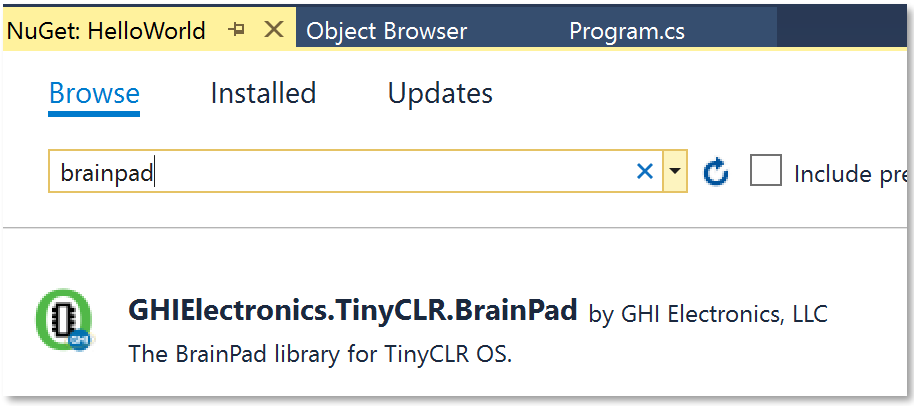
Досі ми використовували «чисту» TinyCLR OS для запуску простих програм. BrainPad поставляється з бібліотеками, щоб допомогти у використанні всіх доступних портів вводу-виводу. Клацніть правою кнопкою миші на проект у вікні Solution Explorer і оберіть «Керування пакетами NuGet» ...



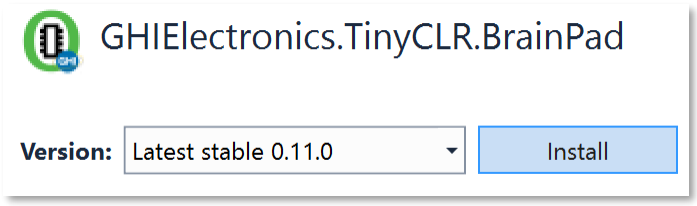
NuGet – це онлайн-сервіс для розміщення бібліотек. Також ви можете завантажувати бібліотеки і розміщати їх локально на своєму ПК. Оберіть шлях до бібліотеки. Наприклад, у нас є бібліотеки, розміщені локально у каталозі з іменем TinyCLR.



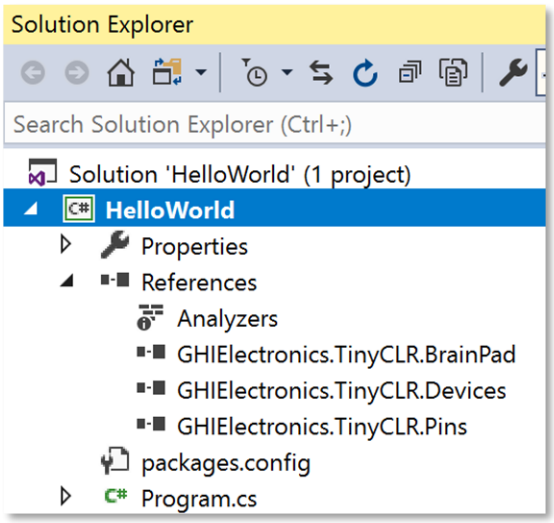
У вкладці Browse наберіть «brainpad» у пошуку.



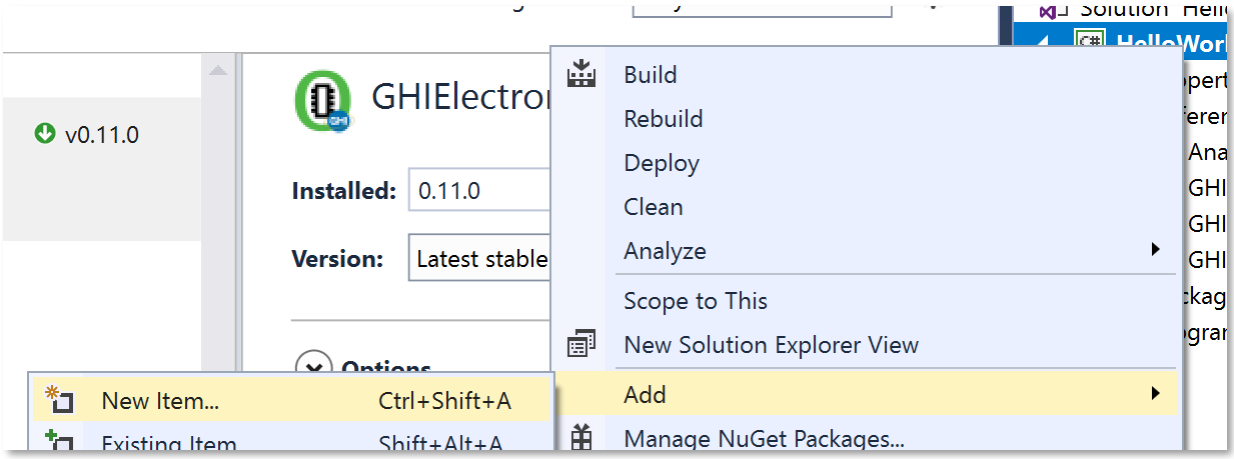
Ви побачите бібліотеку GHIElectronics.TinyCLR.BrainPad. Оберіть її і натисніть «Install».



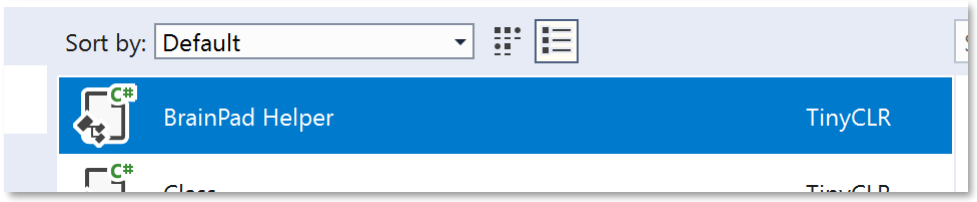
Так ви додасте необхідні бібліотеки по пошуку в Explorer.



Додати бібліотеки у ваш код дуже легко. Для цього ви можете використовувати допоміжний клас. Клацніть правою кнопкою миші по проекту, оберіть «Add > New Item...». Також ви можете використовувати комбінацію клавіш Ctrl + Shift + A.



З доступних опцій оберіть BrainPad і натисніть кнопку «Add».



Зараз почнеться найцікавіше! Перейдіть у вікно Program.cs, де ви написали програму для лічильника раніше. Змініть ваш код, щоби ваш лічильник відображався на дисплеї BrainPad.

```
namespace HelloWorld
{
    class Program
    {
        static void Main()
        {
            var count = 0;

            while (true)
            {
                BrainPad.Display.DrawSmallText(0, 0, "Count: " + count);
                BrainPad.Display.RefreshScreen();
                BrainPad.Wait.Seconds(1);

                count = count + 1;
            }
        }
    }
}
```

Рядок, який відповідає за вивід лічильника, аналогічний команді Debug.WriteLine, яку ми використовували раніше, за виключенням того, що вона виводиться на дисплеї BrainPad. Другий рядок необхідний для оновлення екрану. Дисплеї трохи повільніші, ніж процесори. Якщо ми будемо оновлювати екран кожен раз, коли ми малюємо, дисплей буде працювати дуже повільно. Замість цього процесор звертається до своєї пам'яті. Лише тоді викликається RefreshScreen,

дисплей оновлюється. Уявіть відеогру, де вам потрібно намалювати корабель, перешкоди, ворогів тощо. Ви будете це малювати лише у пам'яті, яка працює швидко і оновлювати дисплей будете лише за морою необхідності.

Останній рядок – попросити BrainPad почекати одну секунду.

М'ЯЧ, ЩО ПІДСТРИБУЄ

Ми вже говорили про те, що практично всі мови програмування використовують англійську мову і, якщо ви її розумієте, то зрозуміти мову програмування вам буде ще легше. Так ось, читати код, написаний за допомогою бібліотеки BrainPad – це ніби як читати цікаву книгу англійською мовою. Додайте деяку логіку і ви зможете створювати дивовижні програми. Викладання C # / Visual Basic не розглядається у цій книзі, але це не означає, що ми не можемо трохи розважитись!

Повернімося до початкової програми і намалюємо круг діаметром 5 пікселів. Центр його помістимо у точці з координатами 30,30. Координати на дисплеях на комп'ютерних системах починаються у верхньому лівому куті і задаються як 0,0. Переміщення фігур направо робиться шляхом збільшення x (першого числа). 30,0 задає положення точки 30 пікселів від лівого краю екрану і 0 пікселів від верхнього краю, тобто у самому верху дисплею.

```
namespace HelloWorld
{
    class Program
    {
        static void Main()
        {
            var x = 30;
            var y = 30;

            while (true)
            {
                BrainPad.Display.DrawCircle(x, y, 5);
                BrainPad.Display.RefreshScreen();
                BrainPad.Wait.Seconds(1);
            }
        }
    }
}
```

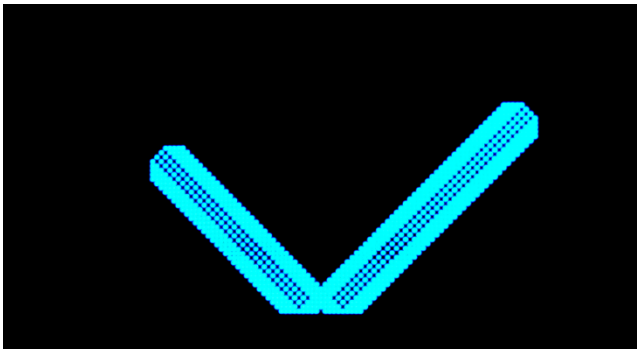
І у нас вийшов круг!

У нашому нескінченному циклі ми будемо збільшувати змінні X і У у кожному циклі. Оскільки ми почали з 30, змінна буде мінятися так: 31,32,33,34 тощо. Все цікавіше і цікавіше, авжеж?

Круг рухається повільно з затримкою в одну секунду. Ми змінимо це значення на 0,01. Також додаймо деяку логіку, щоб коло/шар відскакував назад. Для цього нам потрібні нові змінні DX і DY. Вони будуть зберігати в собі напрямок, в якому рухається м'яч. Потім ми перевіряємо, чи знаходиться коло скраю. Якщо це так, ми змінимо напрямок.

```
namespace HelloWorld
{
    class Program
    {
        static void Main()
        {
            var x = 30;
            var y = 30;
            var dx = 1;
            var dy = 1;

            while (true)
            {
                x = x + dx;
                y = y + dy;

                if (x < 0 || x > BrainPad.Display.Width)
                {
                    dx = dx * -1;
                }

                if (y < 0 || y > BrainPad.Display.Height)
                {
                    dy = dy * -1;
                }

                BrainPad.Display.DrawCircle(x, y, 5);
                BrainPad.Display.RefreshScreen();
                BrainPad.Wait.Seconds(0.01);
            }
        }
    }
}
```

Коло перемальовується під час руху екраном.



Не схоже на шар, який рухається… Проблема у тому, що нам потрібно очищати дисплей кожен раз перед тим, як намалювати нове коло. Додайте цей рядок прямо перед `DrawCircle`.

```
BrainPad.Display.Clear();
```

Ви хочете прискорити його? Затримка 0,01 дуже маленька, тому проблема не в цьому. Ми переміщуємо коло по одному пікселю у кожному циклі. Перемістіть його на 5 пікселів замість цього і він буде у п’ять разів швидше. Це досягається шляхом зміни початкового значення DX і DY.

```
var dx = 5;
var dy = 5;
```

Який шар відскакує і не створює шуму? Використовуйте функцію «Вверх» кожен раз, коли м’яч відскакує. Додайте цей рядок всередині кожного оператора "if".

```
BrainPad.Buzzer.Beep();
```

Ось так! Тепер це м’яч скаче! Але це надто голосно, особливо, якщо ви в класі з тридцятьма BrainPada'ми! Додайте перевірку, щоб м’яч видавав звук лише під час натискання кнопки вниз.

```
if (BrainPad.Buttons.IsDownPressed())
{
    BrainPad.Buzzer.Beep();
}
```

Чи можете ви додати крутий ефект, роблячи м’яч все більшим і більшим під час руху? Подумайте, як змінити положення шара у встановлених межах. Ми будемо робити те ж саме. Нам потрібна змінна R для радіусу і змінна DR для напрямку радіусу, який розширюється чи стискається.

```
var r = 3;
var dr = 1;
```

Далі ми збільшуємо розмір з межами від 1 до 8.

```
r = r + dr;
```

```
if (r < 1 || r > 8)
{
    dr = dr * -1;
}
```

Звичайно, нам треба змінити рядок DrawCircle і використовувати змінну R замість 5.

```
BrainPad.Display.DrawCircle(x, y, r);
```

Ось повний код. Але, перш ніж його копіювати і вставляти, впевніться, що ви назвали свій проект HelloWorld (без пробілів між словами, з врахуванням реєстру), якщо у вас інша назва, тоді треба змінити простір імен (namespace)відповідно до назви вашого проекту, а не копіювати. Наприклад:

```
namespace SomethingElse
```

Потім збережіть цей рядок, але скопіюйте іншу частину коду – все, починаючи від класу Program і закінчуючи другою і останньою правою дужкою (}). Остання дужка насправді є частиною простору імен.

```
namespace HelloWorld
{
    class Program
    {
        static void Main()
        {
            var x = 30;
            var y = 30;
            var dx = 5;
            var dy = 5;
            var r = 3;
            var dr = 1;

            while (true)
            {
                x = x + dx;
                y = y + dy;

                if (x < 0 || x > BrainPad.Display.Width)
                {
                    if (BrainPad.Buttons.IsDownPressed())
                    {
                        BrainPad.Buzzer.Beep();
                    }

                    dx = dx * -1;
                }

                if (y < 0 || y > BrainPad.Display.Height)
                {
                    if (BrainPad.Buttons.IsDownPressed())
                    {
                        BrainPad.Buzzer.Beep();
                    }
```

```
                    dy = dy * -1;
                }

                r = r + dr;

                if (r < 1 || r > 8)
                {
                    dr = dr * -1;
                }

                BrainPad.Display.Clear();
                BrainPad.Display.DrawCircle(x, y, r);
                BrainPad.Display.RefreshScreen();
                BrainPad.Wait.Seconds(0.01);
            }
        }
    }
}
```

РІЗДВЯНІ ВОГНІ

Ви правильно прочитали цей заголовок. Ми будемо створювати різдвяний вогник з одним світлодіодом. Для цього ми будемо використовувати одну з тисячі вбудованих в TinyCLR OS функцій. За допомогою цієї функції ми отримаємо випадкове число. На відміну від реального життя, у комп'ютері немає нічого, дійсно випадкового. Це може ускладнити нашу роботу. Але існує вбудована функція випадкових чисел, яка спрощує нашу задачу. Ми просто створюємо випадковий об'єкт, а потім задаємо йому наступне випадкове значення.

Ми будемо використовувати ці випадкові числа, щоб встановити основні кольори світлодіодів червоного, зеленого і синього. Ви вже знаєте, що ці три кольори можуть створити будь-який колір, який ви хочете.

```
static void Main()
{
    var rnd = new Random();

    while (true)
    {
        BrainPad.LightBulb.TurnColor(rnd.Next(100), rnd.Next(100), rnd.Next(100));
        BrainPad.Wait.Seconds(0.1);
    }
}
```

БАЧИТИ СВІТ

Припустимо, ми хочемо покласти рослину де-небудь в приміщенні, і нам цікаво знати, скільки світла у цьому місці. Ми хочемо побачити на діаграмі, як змінюється рівень освітленості протягом дня.

Діаграма дуже проста. Ми будемо малювати вертикальні лінії, початок яких задається рівнем освітленості, а кінець – у самому низу дисплею. Кожна наступна лінія зсувається на один піксель вправо і з кожною новою ітерацією (повторення дії) в циклі.

```
static void Main()
{
    var x = 300;

    while (true)
    {
        x++;

        if (x > BrainPad.Display.Width)
        {
            x = 0;

            BrainPad.Display.Clear();
            BrainPad.Display.DrawSmallText(30, 0, "Light Level");
        }

        var light = BrainPad.LightSensor.ReadLightLevel() / 2;
        var y = BrainPad.Display.Height – light;

        BrainPad.Display.DrawLine(x, y, x, BrainPad.Display.Height);
        BrainPad.Display.RefreshScreen();
        BrainPad.Wait.Seconds(0.1);
    }
}
```

Ймовірно, ви хочете, щоб програма зупинилася, коли лінії дійшли до правого краю, але у цьому прикладі ми очистимо дисплей і почнемо з самого початку. Давайте швидко запустимо цикл, щоби ми могли бачити результати. Якщо вам потрібен графік освітленості за дві години, а екран має ширину 128 пікселів, вам знадобиться однохвилинна затримка на кожному кроці. Це дозволить записати 128 хвилин по ширині дисплею.

Також ви можете зберегти програму такою, як вона є, а потім накрити датчик освітленості рукою, щоби зробити кілька цікавих малюнків.

Оскільки ви вже експерти, спробуйте дати відповіді на ці питання:

1. Чому ми ділимо показники рівня освітленості на 2?

2. Чому ми встановили початок лінії в BrainPad.Display.Height відповідно до рівня освітленості?
3. Як ви вважаєте, чому X стає активним зі значенням 300? Що відбудеться, якщо замість цього використати 0?

Ось відповіді, але пообіцяйте, що спробуєте самостійно!

Рівень освітленості знаходиться у межах від 0 до 100. Дисплей має висоту 64 пікселя. Якщо ми розділимо рівень навпіл, максимальне значення дорівнюватиме 100/2 або 50. При відображенні на 64 пікселя ми отримаємо 14 пікселів у верхній частині екрану для відображення рівня. Тобто, масштабування.

Відповідь на друге питання полягає в тому, що ми хочемо показати збільшення рівня освітленості. Менші координати У показані у верхній частині дисплею, але ми хочемо, щоб нижні рівні освітленості з'явилися у нижній частині дисплею. Віднімаючи рівень освітленості від висоти дисплею, ми інвертуємо (змінюємо на протилежний) графік.

Відповідь на третє питання. Встановити на старті велике значення X – це хороший трюк, який допоможе під час першого запуску коду. Погляньте на програму уважно. Ми говоримо програмі, що дисплей уже заповнений показниками рівня освітленості і його треба очистити. Тобто, від самого початку програми виконається умова «if», дисплей очиститься і на ньому відобразиться надпис «Light Level», що загалом буде відбуватися після кожного заповнення екрану показниками рівня освітленості. Спробуйте змінити його до 0, і показники датчика не будуть відображатися на екрані до першого заповнення дисплею. Ви можете просто очищати дисплей після кожної ітерації, але це уповільнить вашу програму. Ми завжди повинні мислити як програмісти, коли ми пишемо програму і робити тільки те, що необхідно.

БАГАТОПОТОКОВІСТЬ

Просити систему зробити кілька речей одночасно дуже складно. Як правило, це важко для користувачів невеликих систем. Але є хороша новина: TinyCLR OS володіє функціоналом багатопотоковості і її легко використовувати. Це працює точно так, як для звичайної системи, яка працює в середовищі .NET. Багатопотоковість називається threading, де кожен потік є окремою задачею, що виконує частину програми.

Невпевнені, чи потрібно використовувати потоки? Як би ви написали програму, яка одночасно виводить на дисплей значення лічильника і блимає світлодіодом? Звичайно, якщо ви хочете збільшувати лічильник з кожним блиманням, тоді це легко.

```
static void Main()
{
    var count = 0;
```

```
    while (true)
    {
        BrainPad.LightBulb.TurnGreen();
        BrainPad.Wait.Seconds(0.1);
        BrainPad.LightBulb.TurnOff();
        BrainPad.Wait.Seconds(0.9);

        BrainPad.Display.DrawSmallText(0, 0, "Count: " + count);
        BrainPad.Display.RefreshScreen();

        count = count + 1;
    }
}
```

Тепер змініть програму, щоб світлодіод блимав один раз на секунду, а значення лічильника змінювалось якомога швидше. Ви не обійдетеся без потоків. Почніть з переміщення коду для блимаючого світлодіоду у свій власний метод. Метод – це фрагмент коду, який виконує певну задачу. Ви викликаєте цей метод для обробки цієї задачі. Наприклад, LightBulb має метод TurnGreen, який запалює його зеленим кольором.

```
static void Blink()
{
    while (true)
    {
        BrainPad.LightBulb.TurnGreen();
        BrainPad.Wait.Seconds(0.1);
        BrainPad.LightBulb.TurnOff();
        BrainPad.Wait.Seconds(0.9);
    }
}
```

Якщо ви скористаєтесь цим методом, він буде блимати один раз на секунду, але він ніколи не вийде зі свого циклу while. Це нормально, тому що ми будемо викликати цей метод з окремого потоку. Чи готові ви створити складний код для потоку? Ось він!

```
new Thread(Blink).Start();
```

Ми просто сказали системі створити новий потік для методу Blink і запустити його.

Стоп! Перед запуском коду нам потрібні останні зміни. У кожному потоці в системі повинна бути затримка. Це допомагає в обробці внутрішніх задач і дозволяє іншим потокам працювати правильно. Але ви можете сказати, що у нас лише один потік. Насправді, у нас два потоки. Система завжди створює внутрішній потік, який запускає метод Main () (основний метод запуску). Якщо вам треба, щоб програма працювала якомога скоріше, просто додайте мінімально затребувану затримку.

```
 BrainPad.Wait.Minimum();
```

Тут обидва методи – Main і Blinking.

```
static void Blink()
{
```

```
    while (true)
    {
        BrainPad.LightBulb.TurnGreen();
        BrainPad.Wait.Seconds(0.1);
        BrainPad.LightBulb.TurnOff();
        BrainPad.Wait.Seconds(0.9);
    }
}

static void Main()
{
    new Thread(Blink).Start();

    var count = 0;

    while (true)
    {
        BrainPad.Display.DrawSmallText(0, 0, "Count: " + count);
        BrainPad.Display.RefreshScreen();
        BrainPad.Wait.Minimum();

        count = count + 1;
    }
}
```

Існує два нескінченних цикли і обидва вони працюють одночасно. Хоча це дуже простий приклад, але він показує можливості. Щоб потренуватися, додайте ще один потік для подачі звукового сигналу кожні три секунди.

ЗАТЕЛЕФОНУЙ МЕНІ

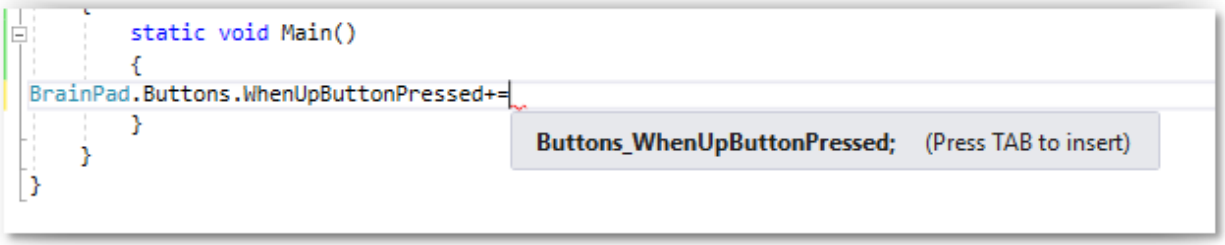
Ми вже використовували кнопки, і ви знаєте, як вони працюють. Однак, ми завжди використовували їх у циклі, де перевіряли, чи була натиснута кнопка. Система може перевірити мільйони разів, чи натиснута кнопка, перш ніж вона буде натиснута. Це дуже погано для приладу з живленням від батареї. Добре написана програма іде у сплячий режим, коли їй не треба обробляти жодних задач. Це робиться для економії заряду батареї. Якби ваш телефон не «засинав», то не пропрацював би і години без підзарядки.

Ця тема досить широка, тому ми розглянемо лише події для кнопок. Замість постійної перевірки ми попросимо систему «зателефонувати» нам під час натискання. Ми хочемо, щоб кожен раз під час натискання кнопки «вверх» BrainPad видавав звуковий сигнал. А в циклі нехай блимає світлодіод.

```
static void Main()
{
    while (true)
    {
        BrainPad.LightBulb.TurnGreen();
        BrainPad.Wait.Seconds(0.1);
        BrainPad.LightBulb.TurnOff();
        BrainPad.Wait.Seconds(0.9);

        if (BrainPad.Buttons.IsUpPressed())
```

```
        {
            BrainPad.Buzzer.Beep();
        }
    }
}
```

Натисніть кнопку «вверх» і нічого не відбудеться. Утримуйте її і BrainPad буде подавати звуковий сигнал один раз на секунду. Це відбувається, оскільки цикл має затримку і кнопка перевіряється один раз на секунду. Ви можете зменшити затримку, але це не вирішить проблему. Ми підемо іншим шляхом і будемо використовувати події. Visual Studio може автоматично генерувати необхідний код. Почніть із вводу «BrainPad.Buttons.WhenUpButtonPressed», а потім додайте «+ =». Тепер ви можете натиснути клавішу табуляції, щоб згенерувати метод події.



Згенерований метод події виглядає наступним чином:

```
private static void Buttons_WhenUpButtonPressed()
{
    throw new NotImplementedException();
}
```

Згенерований код має рядок для обробки помилок (винятки). Можна вважати, що це нагадування, щоби ви не забули заповнити даний метод нашим кодом. Замініть цей рядок на звуковий сигнал.

```
private static void Buttons_WhenUpButtonPressed()
{
    BrainPad.Buzzer.Beep();
}
```

Не забудьте видалити код, який перевіряє кнопку в основному циклі.

```
static void Main()
{
    BrainPad.Buttons.WhenUpButtonPressed += Buttons_WhenUpButtonPressed;

    while (true)
    {
        BrainPad.LightBulb.TurnGreen();
        BrainPad.Wait.Seconds(0.1);
        BrainPad.LightBulb.TurnOff();
        BrainPad.Wait.Seconds(0.9);
    }
}

private static void Buttons_WhenUpButtonPressed()
```

```
{
    BrainPad.Buzzer.Beep();
}
```

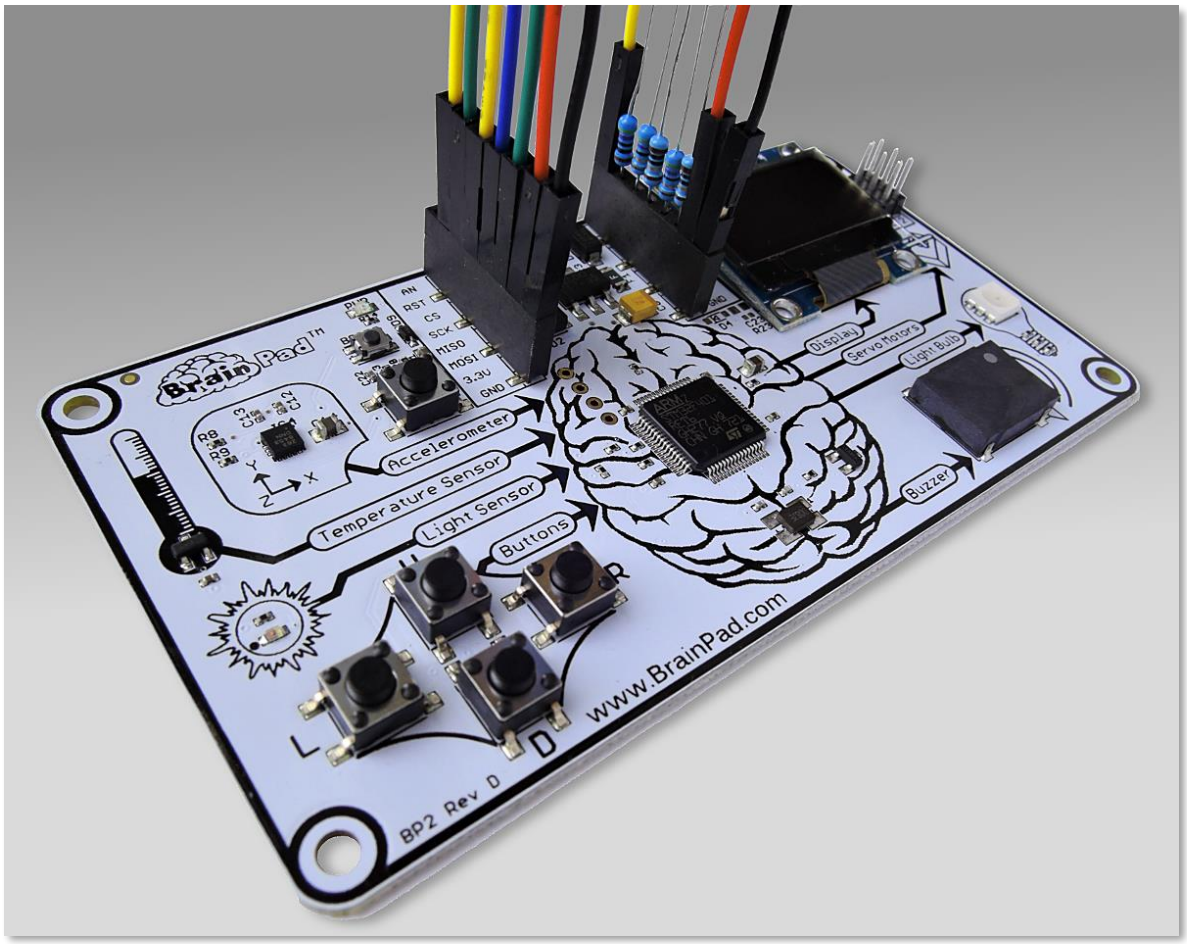
До речі, кілька подій можуть викликати один і той самий обробник подій. Ось код, який подає звуковий сигнал під час натискання будь-якої з чотирьох кнопок.

```
static void Main()
{
    BrainPad.Buttons.WhenUpButtonPressed += Beeper;
    BrainPad.Buttons.WhenDownButtonPressed += Beeper;
    BrainPad.Buttons.WhenLeftButtonPressed += Beeper;
    BrainPad.Buttons.WhenRightButtonPressed += Beeper;

    while (true)
    {
        BrainPad.LightBulb.TurnGreen();
        BrainPad.Wait.Seconds(0.1);
        BrainPad.LightBulb.TurnOff();
        BrainPad.Wait.Seconds(0.9);
    }
}

private static void Beeper()
{
    BrainPad.Buzzer.Beep();
}
```

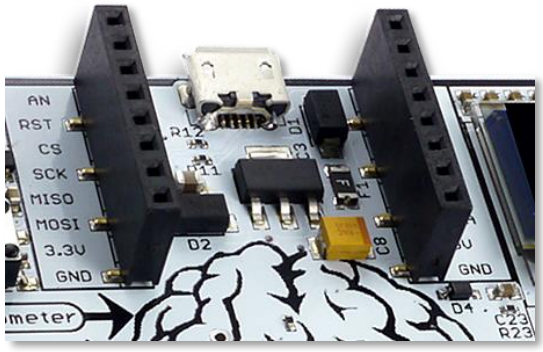
ДРОТИ, ЩО ЛЯКАЮТЬ

Це звичайна справа, коли новачки в електроніці бояться навіть думки, про те, щоб підключитись до чогось дротами. Що буде, якщо станеться коротке замкнення чи прилад «згорить»?

Не хвилюйтесь з цього приводу! BrainPad готовий до важких випробувань. По-перше, він працює з дуже низькою напругою в 5В. Ви не зможете отримати травму, доторкнувшись до 5В. Крім того, BrainPad має вбудований захист і призначений для того, щоб його було важко пошкодити, під’єднавши щось неправильно. Тим не менш, ви повинні розуміти, що робите. Існує багато курсів і онлайн-статей, які допоможуть вам зрозуміти схеми.

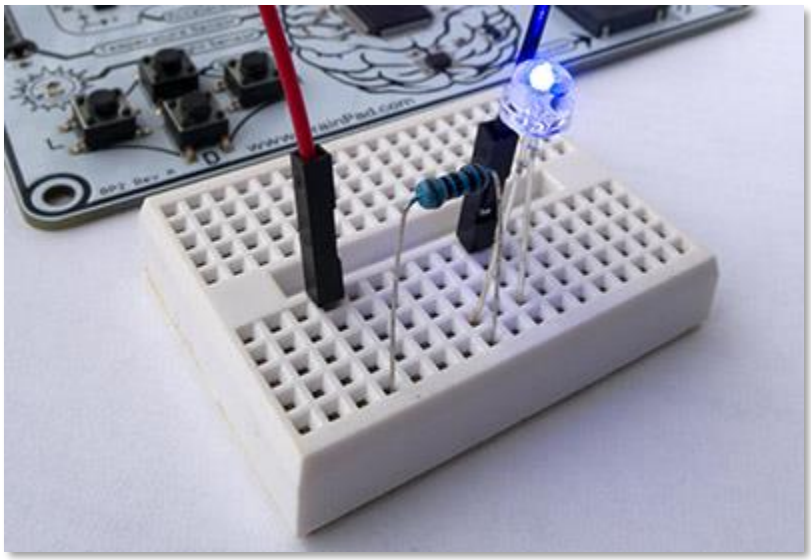


Як згадувалось в першому розділі цієї книги, існує кілька способів розширення можливостей BrainPad завдяки коннекторам (expansion connectors),розміщених поруч з роз'ємом USB.

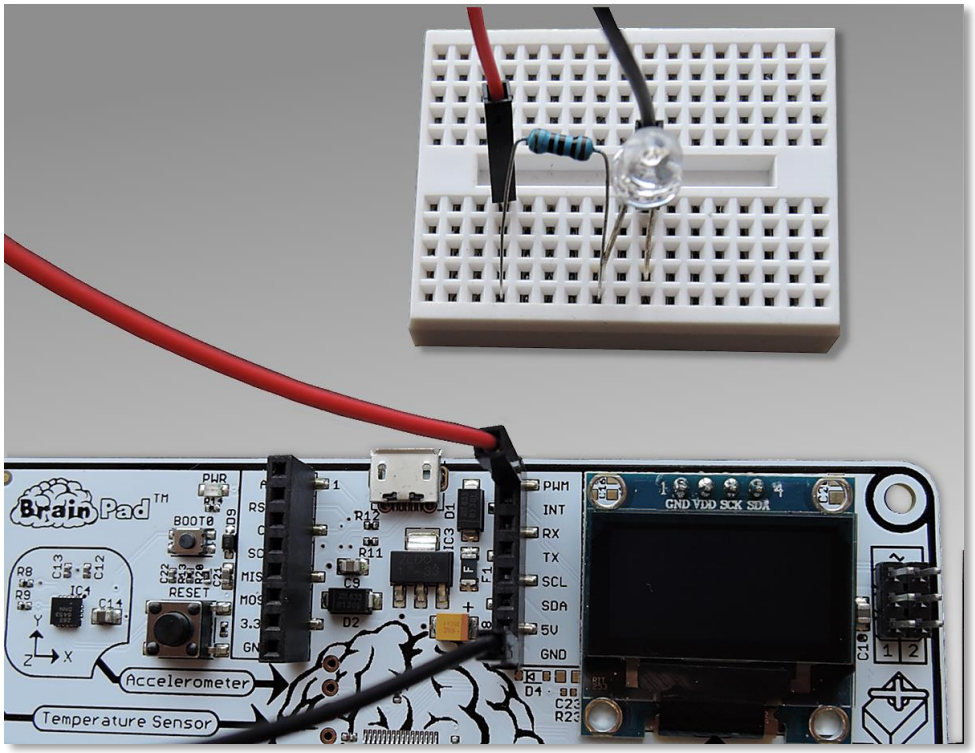


Ці роз'єми мають багато функцій і нам потрібно кілька книг, щоб повністю описати їх. У цій книзі ми наведемо простий приклад, підключивши світлодіоди (LED).

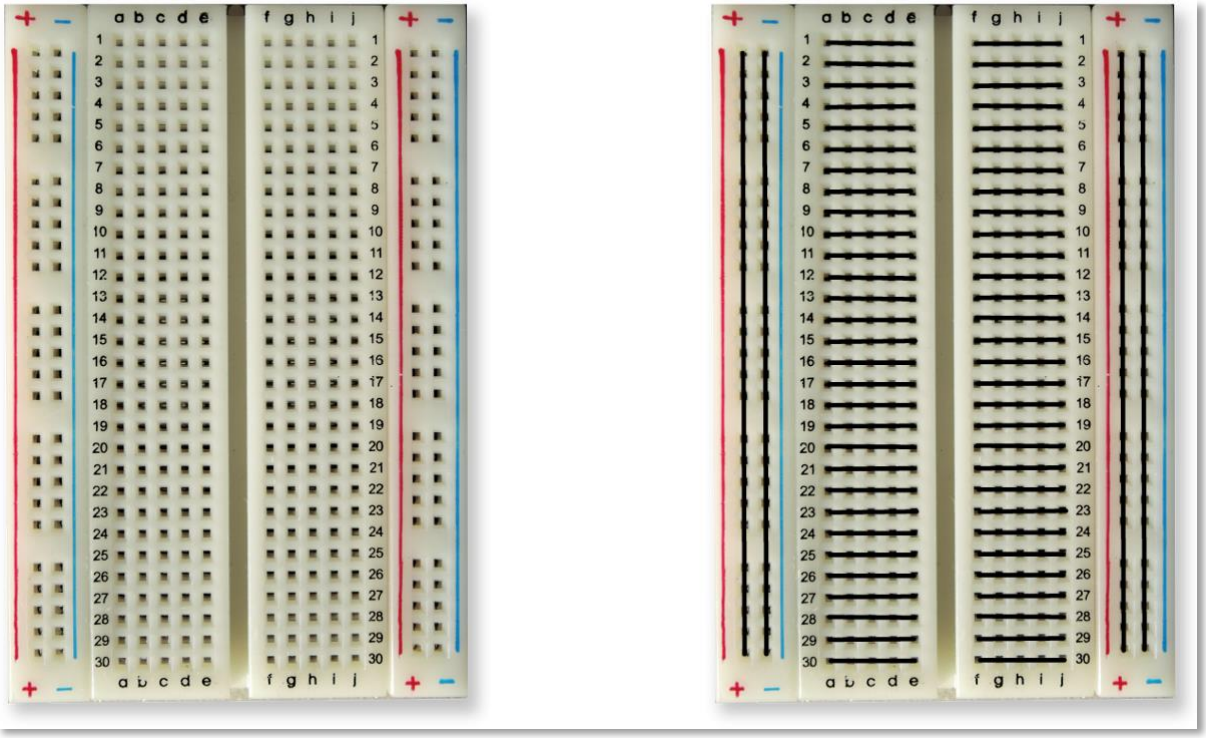
LED дешеві і мають різні кольори. У них є два контакти, один з яких довший за інший. Деякі світлодіоди мають більше одного кольору (наприклад, LED BrainPad). Для керування світлодіодом необхідний струмообмежуючий резистор. Деякі розповсюджені номінали: 220, 330 або 470 Ом. Ще вам знадобляться дроти і макетна дошка (bread-board).



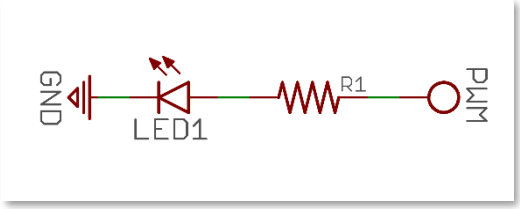
Ми можемо використовувати один з контактів GPIO в expansion connectors для керування світлодіодом. GPIO – означає висхідний сигнал загального призначення. Ми будемо використовувати контакт з маркуванням PWM.



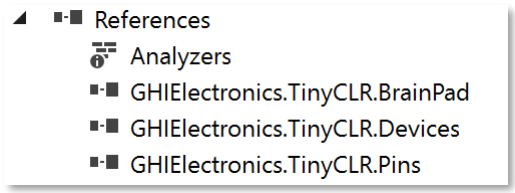
Bread-board має внутрішні з’єднання, де кожен ряд отворів замкнутий. На зображенні нижче bread-board з правого боку має чотири лінії, які показують внутрішні електричні з’єднання. Усе, що підключене до отворів, з’єднаних чорною лінією, буде під’єднане між собою.



Дріт буде з'єднувати PWM-контакт з контактом на bread-board, куди під'єднаний довгий дріт від світлодіоду. Короткий контакт світлодіоду повинен бути підключений до резистора, а резистор – до контакту, що заземляє. Це утворює ланцюг (або схему), де електрони будуть рухатися між PWM і GND. Коли PWM-контакт активований, загориться світлодіод.



Для керування виводом PWM, який є GPIO, нам потрібно додати бібліотеки GHIElectronics.TinyCLR.Devices і GHIElectronics.TinyCLR.Pins з NuGet. Ці бібліотеки використовує бібліотека GHIElectronics.TinyCLR.BrainPad, які ми завжди додаємо, тому жодних додаткових дій не потрібно, оскільки вони автоматично завантажуються.



Нам знадобиться доступ до бібліотек GPIO у нашій програмі. Це робиться за допомогою одного рядка коду.

```
using GHIElectronics.TinyCLR.Devices.Gpio;
```

Нам треба отримати доступ до контакту з засобів контролера GPIO, який знаходиться всередині процесора.

```
var controller = GpioController.GetDefault();
var pwmPin = controller.OpenPin(BrainPad.Expansion.GpioPin.Pwm);
```

Контакти (піни) можуть мати функцію, як виходу, так і входу точно так же, як виходи і входи на BrainPad. Входи подають сигнали у «мозок», а виходи – з «мозку». Приклад вводу – це кнопка, наприклад, кнопки на BrainPad. Прикладом виводу може бути світлодіод. Таким чином, контакт PWM повинен бути вихідним.

```
pwmPin.SetDriveMode(GpioPinDriveMode.Output);
```

Тепер можемо встановити (write) рівень на контакті. Рівень може бути високим (high), що означає, що контакт активований, або низьким (low) – контакт відключений. Помістіть це в цикл з деякими паузами. Результатом буде блимаючий світлодіод.

```
while (true)
{
    pwmPin.Write(GpioPinValue.High);
    BrainPad.Wait.Seconds(0.1);
    pwmPin.Write(GpioPinValue.Low);
    BrainPad.Wait.Seconds(0.5);
}
```

Повністю програма повинна виглядати ось так:

```
static void Main()
{
    var controller = GpioController.GetDefault();
    var pwmPin = controller.OpenPin(BrainPad.Expansion.GpioPin.Pwm);

    pwmPin.SetDriveMode(GpioPinDriveMode.Output);

    while (true)
    {
        pwmPin.Write(GpioPinValue.High);
        BrainPad.Wait.Seconds(0.1);
        pwmPin.Write(GpioPinValue.Low);
        BrainPad.Wait.Seconds(0.5);
    }
}
```

Підключення кнопки аналогічне, за виключенням того, що вивід повинен бути входом замість виходу. Не вдаючись у подробиці, додамо, що контакт повинен бути входом з підтягуванням. Це утримує пін на рівні high доти, доки не буде натиснута кнопка.

```
pwmPin.SetDriveMode(GpioPinDriveMode.InputPullUp);
```

Не потрібно додавати резистори до кнопки. Просто підключіть кнопки між одним з GPIO і GND-контактами.

ВИСНОВКИ

TinyCLR OS від GHI Electronics пропонує дійсно професійний рівень програмування для BrainPad. Знання, які ви отримаєте , будуть вам корисні для програмування будь-якого пристрою – від телефону до комп’ютера. І все це спрощується завдяки використанню TinyCLR Operating System і Microsoft .NET Framework.

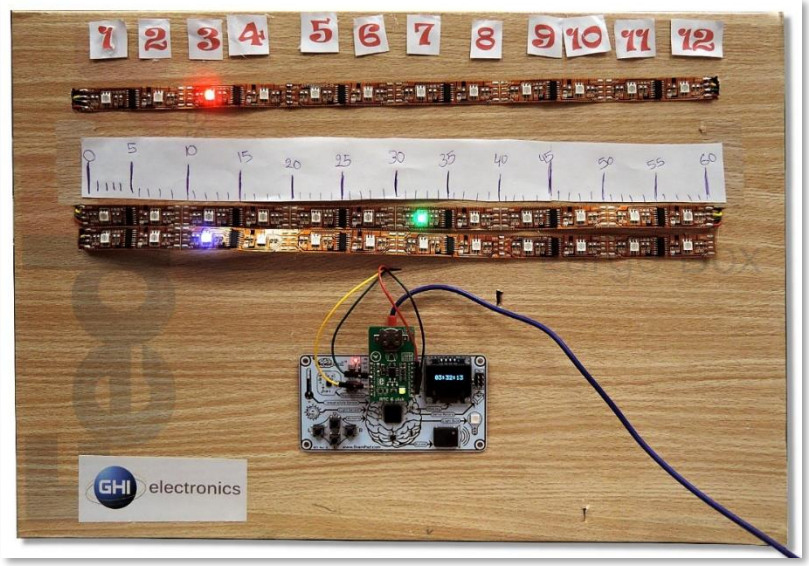
РОЗШИРЕННЯ МОЖЛИВОСТЕЙ

Додаткові контакти на BrainPad відкривають цілий світ можливостей. Тут ми перелічимо деякі варіанти, щоби ви могли подумати про те, що можливо.

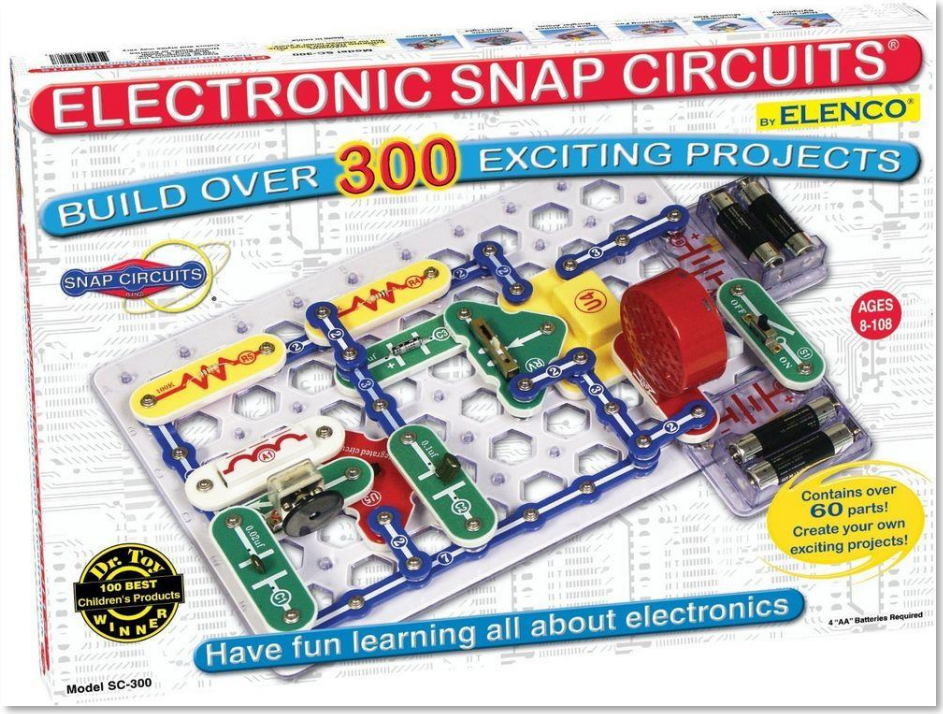
MIKROELEKTRONIKA CLICK BOARDS™

MikroElektronika пропонує сотні невеликих модулів (click module), які підключаються напряму до BrainPad. Модулі включають в себе прості датчики, наприклад, для вимірювання вологості і температури. Існують і складніші пристрої, такі як розпізнавання голосу і модулі електрокардіограми. Існує також безліч керованих модулів для керування двигуном і цифровим освітленням. Ви зможете знайти їх на цьому веб-сайті https://www.mikroe.com/click.

У цьому прикладі проекту ми створили лінійний годиннке, який використовують click module з функцією (RTC), щоби відслідковувати час навіть коли BrainPad не підключений (https://docs.brainpad.com/projects/linear-clock.html).



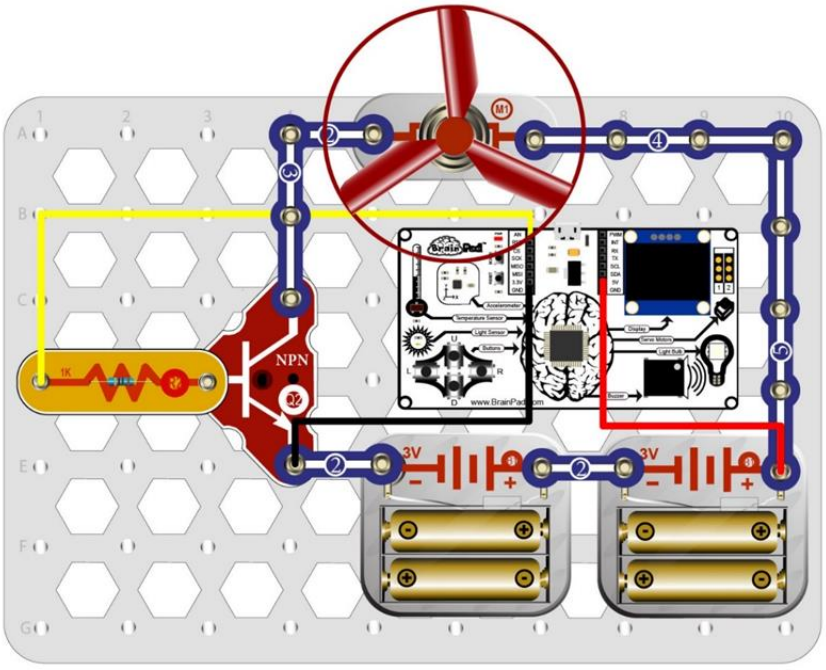
ELENCO® SNAP CIRCUITS®



Одна із найпопулярніших платформ для вивчення електроніки називається Snap Circuits (виробництво Elenco (https://www.elenco.com)). Ці популярні набори можна знайти в інтернет-

магазинах для радіоаматорів. Включають вони в себе різні електронні компоненти й інструкції для створення безлічі різноманітних проектів.

З BrainPad ви можете додати інтелект і запрограмувати свої проекти з електроніки. Цей приклад проекту представляє собою таймер зворотного відліку, що запускає пропелер. Рекомендуємо

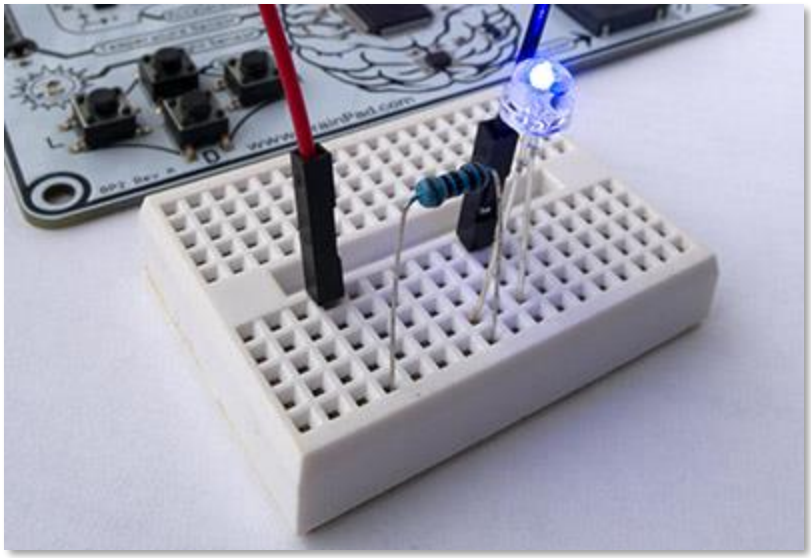


подивитись відео. Це один із наших фаворитів (https://docs.brainpad.com/projects/lift-off.html).

BREADBOARDS

Бажаєте проектувати свої електронні схеми з нуля так, як це роблять професійні інженери? Існує багато способів створення схем, які працюють з BrainPad. І хоча проектування схем потребує більше знань, це дуже творчий і найдешевший спосіб розширити можливості вашого BrainPad. І це шлях, який принесе вам більше всього знань.

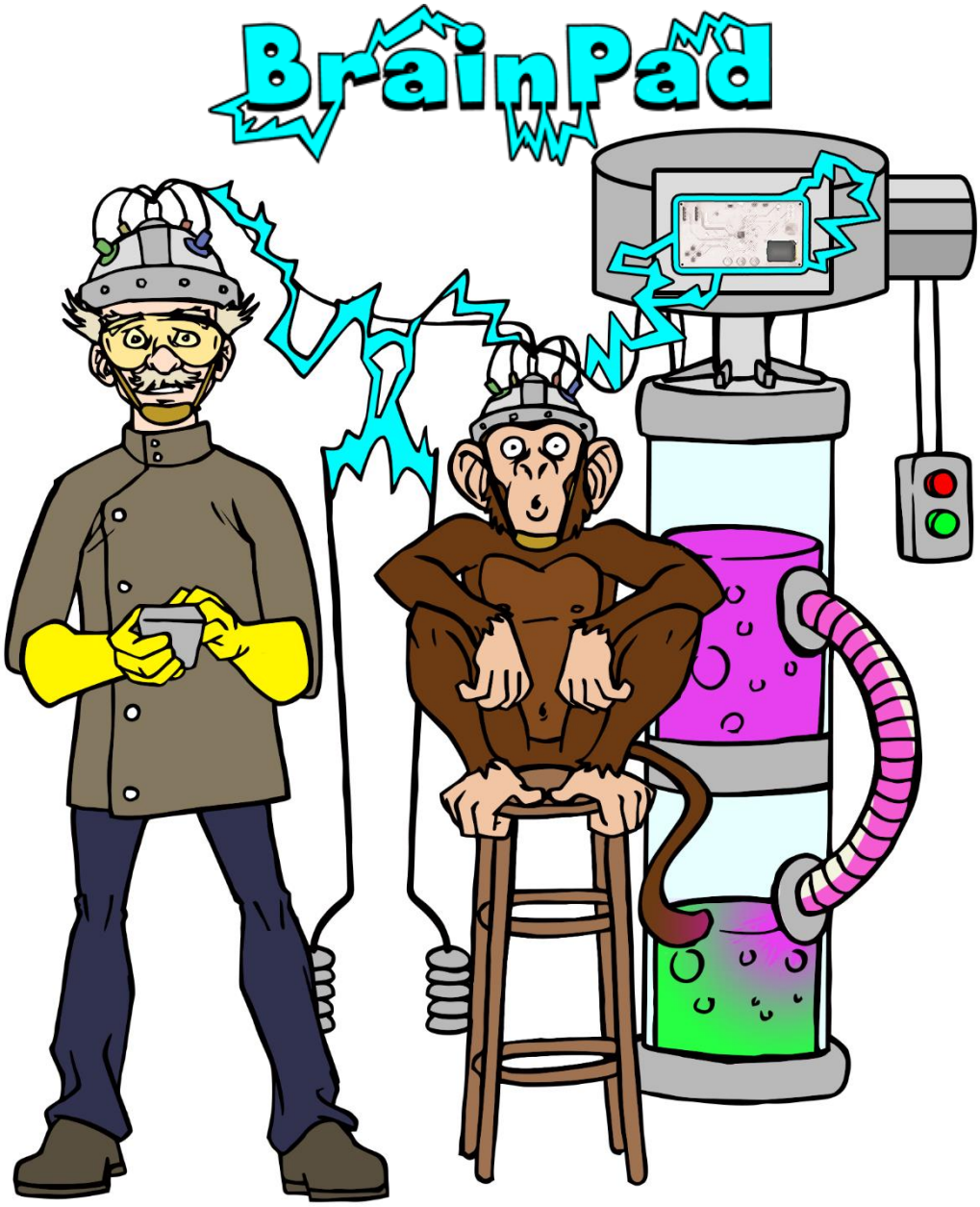
Найпростіший спосіб створювати власні схеми – використовувати bread-boards макети. Дроти і компоненти просто вставляються в отвори. Макет містить компоненти і з’єднує їх разом електрично. Крім того, це найшвидший спосіб протестувати нову схему і виправити допущені помилки.



Якщо б ви хотіли використовувати певний макет постійно, є кілька варіантів: від простих – попередньо виготовивши прототип, до складніших – проектування та виготовлення ваших власних друкованих плат. Креслення друкованих плат можна виготовити вручну або за допомогою безкоштовних комп'ютерних інструментів для проектування. Це те, що робить BrainPad таким особливим – з ним можна легко почати і просунутися так далеко, наскільки вам подобається і при цьому працювати у зручному для вас темпі.

Також ви можете додати до вашого проекту потрібні вам датчики. Пошукайте в Інтернеті «sensor kit» і ви знайдете багато різних варіантів. Наприклад, ось один:

Для використання цих модулів необхідні певні знання, але у нас є кілька простих проектів, які допоможуть вам рухатися у правильному напрямку. Щойно ви познайомитеся з одним типом сенсора, стане набагато легше зрозуміти, як працюють інші датчики.



www.BrainPad.com