Project Guide

Micrcocontroller Programming

Learning Goals and Skills:

Students learn basic programming logic using Javascript to program microcontrollers. They learn basic circuitry, how digital and analog sensors work, and how to integrate everything into a working project. They learn how computer languages are stuctured and how to use an API. Most importantly, they demystify computers and electronics, discovering that they can build and control them.

Facilities and materials at a glance:

- Some windows, mac, or linux computers (old ones are fine). 2-3 students per computer is best
- Internet connection (optional)
- Microbit microcontrollers ($15 each with edu discount) – one for each group of 2-3 students is ideal
- Microbit breakout boards (optional)
- Breadboards, sensors, servos, wire, etc
- Cheap earbuds or headphones (one per Microbit)
- Recycled art materials (cardboard, paper, plastic bottles…)
- See detailed materials and tools list at end of article

Why use the Microbit?

- Easy to use – ideal for students with no programming experience
- Works offline on just about any computer
- Costs $15 in bulk with edu discount
- Has a host of sensors built-in, so it can do a lot of interesting things standalone
- Has enough capabilities to interest advanced students

- No drivers or software to install- works in any browser

Preparation

In developing these lessons I drew on Microsoft's Intro to CS with MakeCode for Microbit and a bunch of online tutorials. I spent a few days playing with the Microbit to see what I could get it to do.

In a nutshell, you write code at https://makecode.microbit.org, save the code to your Microbit via the usb cable (your computer sees the it as a USB drive), and your code runs on the Microbit, even after you unplug the USB cable.

Given the quality of the documentation out there and the Microbit's ease of use, an enthusiastic beginner could learn enough to teach this project in a few weeks.

About the Program and the Students

We were invited to Louis Trichardt, Limpopo, South Africa by Sumbandila.org, a truly effective organization dedicated to improving the lives of underpriveleged children in South Africa. Funding was provided by Riverdale Country School, where I work. For this project we had five groups of 15-25 students ages 12-16. We saw them for either one or two 45-60 minute periods per day for six days – each group had about 9 hours of class. Most of the students had some experience with computers (Microsoft Word, using a browser), though many had limited typing skills. Knowledge of cutting and pasting, for instance, was not universal. Their English, as a second or third language, ranged from fluent to extremely basic. None of the students had ever done any programming.

Introduction to the Project

The essence of programming is breaking up a process into discrete steps that a computer can understand. I like to introduce programming by asking the students to instruct me, a computer, on how to brush my teeth. Invariably I end up with toothpaste all over my face and on the table before they get it right. “Squeeze the toothpaste” makes a mess, “squeeze the toothpaste tube gently between your left thumb and forefinger for one second” is more useful.

I thought I’d compare a computer program with a recipe, a list of step-by-step instructions where the order matters, but while many had cooked, none of the students had used a recipe before. Lesson: know your audience.

Have the students form groups, ideally in twos if you have enough microcontrollers and computers. Hand out the Microbits. The Microbits come in little boxes with everything needed to get started. As soon as the students put in the batteries the board starts a little demo that introduces its capabilities, engaging the students right away. Once they get to the end of the demo we went over the parts of the board- leds, sensors, pins.

Programming Challenges

Next, we started on a series of fun challenges meant to help students get familiar and confident with programming. There’s a balance to be struck between showing them the code and having them figure it out. As students progress through the challenges, the balance shifts towards the latter. It’s important that when you introduce a new function to let the students try and figure out how it works using the Makecode page – mousing over commands and parameters can yield a lot of clues, as can the dropdown lists that appear when you type a period after an object (e.g. typing “basic” followed by a period gets you a list of its functions: clearScreen, forever, plotLeds, etc.)

These challenges are starting points. As students got comfortable with the Makecode page many started experimenting on their own.

Do you teach programming using draggable blocks or actual code? The basic Microbit web interface - https://makecode.microbit.org - supports both. I went with code because I'm more comfortable with it and think it leads to a deeper understanding of how things work.

For groups of more than a few students a projector is really useful.

Display your name
- Control the LED matrix using showString()
- Definition of a String

```
basic.showString("Kenda")
```

Congratulations! You've just written a program!

Display a heart
- Control the LED matrix another way
- Begin to understand notion of *functions* taking *parameters*

```
basic.showIcon(IconNames.Heart)
```

Display your name over and over again
- Introduce the forever loop

```
basic.forever(() => {
    basic.showString("Kenda")
})
```

Display your name using a variable
- Show the code below and ask students what they think it will do

- Think like a computer – step by step

```
let myName = "Kenda"

basic.showString(myName)
```

What's so great about variables?

- They can change
- Let's try using a sensor
- Explain *input* (info from sensors) vs *output* (led matrix, so far)
- Make temperature change by warming up the board with hands

```
let temp = 0

basic.forever(() => {

    temp = input.temperature()

    basic.showNumber(temp)

})
```

Let's make noise

- Hook up headphones with alligator clips to GND and pin 0
- Makecode page will show connection diagram
- Look at the *parameters* for the playTone *function* – what do they do?
- Once this program is working, ask: how to make the tone change?

```
basic.forever(() => {

    music.playTone(440, 1000)

})
```

This noise is annoying and boring – let's make it change

- Variables are a good way to make things change
- How do you get a variable to change? How do numbers change?
- Lead students to come up with adding 1 to a variable
- How would you do that over and over again? (forever loop)

- What would the line of code look like?
- Try and guide students until they come up with *x = x + 1*
- Once they've got it working, tinker with it to make different sounds
- Walk around with a battery powered speaker and connect to students'
  boards so they can show the rest of the class their ouput

```
let tone = 100

basic.forever(() => {

    music.playTone(tone, 1000)

    tone = tone + 10

})
```

Make a musical instrument using the buttons

- We made tone increase using tone = tone + 10. How to decrease?
- Note use of curly brackets { }. They are like the pieces of bread in a
  sandwich – the beginning and end of a *function body*
- Code within a function's curlies is only executed when the function
  is called
- Functions that begin with "on" are *event handlers* – they are called
  when something happens, like a button press

```
let tone = 500

input.onButtonPressed(Button.A, () => {

    tone = tone - 10

})

input.onButtonPressed(Button.B, () => {

    tone = tone + 10

})

basic.forever(() => {

    music.playTone(tone, 500)
```

```
})
```

Let's make a better musical instrument

- Explore the rotation sensor. How can we control sound with it?
- Have students look at rotation code – mouse over parameters
- It looks like input.rotation *returns* a number – how can we figure out
  what these numbers will look like (their range)?
- How about printing the value to the screen? Print some spaces
  so we can tell the numbers apart

```
let tone = 0

basic.forever(() => {

    tone = input.rotation(Rotation.Roll)

    basic.showNumber(tone)

    basic.showString("  ")

})
```

- It looks like the range returned by input.rotation is -135 to 135.
- Negative pitches don't make a sound (and pitches < 20 are inaudible)
- What can we do to our tone variable so every value makes a sound?

```
let tone = 0

basic.forever(() => {

    tone = input.rotation(Rotation.Roll)

    tone = tone + 155

    music.playTone(tone, 50)

})
```

Can anyone make a siren?

- What has to happen to the variable tone to make a siren sound?
- Let students figure out if has to go up, then down, then up…
- Introduce the *while* loop

```
let tone = 0

basic.forever(() => {

    while (tone < 1000) {

        music.playTone(tone, 1)

        tone = tone + 10

    }

    while (tone > 0) {

        music.playTone(tone, 1)

        tone = tone - 10

    }

})
```

If statements – happy/sad machine

- Have students come up with real-world *if* examples, e.g. if hungry, eat
- Note the punctuation used in programming – parentheses and curlies

  Also note the use of *else:*

  if (hungry) { eat }

  else { sleep }
- Using the rotation sensor and an *if* statement, have the Microbit smile

  when tilted to the right, and frown when tilted to the left

```
let angle = 0

basic.forever(() => {

    angle = input.rotation(Rotation.Roll)

    if (angle > 0) {
```

```
        basic.showIcon(IconNames.Happy)
    }
    else {
        basic.showIcon(IconNames.Sad)
    }
})
```

Further challenges

Given our time frame (six days, one or two 55 minute classes with each group of 20 students per day), it seemed time to introduce the (intentionally vague) final project – to build something that uses a microbit that does something – and to keep working on skills needed for the projects the students chose. If we'd had more time, challenges might include using nested loops and arrays, and defining functions.

Introducing the Final Project

As usual, we set up a public presentation at the project's end – a project fair that would be attended by students, staff, and guests (and a photographer). The photos would be posted on a Riverdale website that all the students back in New York could see. The fair proved to be a great motivator.

What can a microbit do? I showed them how to connect the Microbit to a breakout board + breadboard and how to use jumper wires to attach sensors, leds, and buzzers (although the buzzers I brought didn't work – too much current draw). We talked about the **input sensors** we had to work with:

- Buttons A, B, and AB (both at once)
- Light sensor
- Shake sensor
- Accelerometer
- Temperature sensor

- Rotation sensor
- Compass
- Gestures
- Bluetooth radio receiver (listen for transmission from other Microbits)
- GPIO pins -> circuit open or closed (digitalRead)
- GPIO pins -> motion sensor (digitalRead)

And the **output devices**:

- LED matrix
- Sound (via headphones or computer speakers)
- Bluetooth radio transmitter (send to other Microbits)
- GPIO pins -> servo motor (servoWrite)
- GPIO pins -> standalone LEDs (digitalWrite)
- GPIO pins -> buzzer (ours turned out not to work)

Then I showed some examples I'd found online and we had a brainstorming session. There are a lot of possibilities!

Ideas:

- calculator- inputs are buttons A, B, AB, and shake sensor
- dance alarm (mounted in hallway, detects movement and orders person to dance via screen)
- pedometer (accelerometer)
- dance move counter (similar to pedometer)
- air guitar instrument (sounds and LEDs)
- intruder alarm (copper tape and GPIO circuit)
- animal that moves to its own music? (use servos)
- ball in cup game (balls close GPIO circuit when they land in cups)
- space invaders (LED matrix)
- personal communicator (bluetooth radio)
- piano (GPIO pins attached to keys)
- snake game (LED matrix)

- automatic railroad crossing gate (train closes GPIO circuit)
- carpenter's level (rotation sensor)
- camping compass (compass sensor)

Building the Projects

We spent the remaining days preparing projects for the fair. The students took over. The teachers (myself and two assistants) became facilitators at this point- answering questions, helping students debug and troubleshoot, showing them how to find breaks in copper tape circuits using a multimeter, suggesting ways to make things work. Students came in at lunch and after school into evenings to work on their projects. Their energy was inspiring, the pace exhausting. By fair time we had about 40 projects, most of them working. The students judged it a big success. For photos see lindylabs.org. For questions or comments, jmerrow@riverdale.edu.

Doing this project without an internet connection

Go to https://makecode.microbit.org/ and save the page as HTML. Copy the html file and its companion folder to a flash drive. This can be copied to any offline computer for use with the Microbit.

Detailed Materials List

A simpler version of this project could be done without external circuits – no breakout boards, breadboards, etc. In this case, all you'd need would be the Microbits, headphones, alligator clips and spare batteries. I opted for the breadboard route because I wanted students to experience circuit prototyping. I also wanted to leave a working electronics lab at the school.

Adafruit.com

Products

---

65 x BBC micro:bit Go Bundle[ID:3362] = $1,072.50

65 x Half-size breadboard[ID:64] = $292.50
20 x Premium Male/Male Jumper Wires - 40 x 6" (150mm)[ID:758] = $71.20
20 x Small Alligator Clip to Male Jumper Wire Bundle - 12 Pieces[ID:3255] = $143.20
20 x Premium Female/Male 'Extension' Jumper Wires - 20 x 6"[ID:1954] = $35.20
15 x PIR (motion) sensor[ID:189] = $134.40
15 x Breadboard-Friendly 3.5mm Stereo Headphone Jack[ID:1699] = $12.90

Sub-Total: $1,761.90
Discount Coupon: ADAEDU : -$176.19
United Parcel Service (1 pkg x 37.60 lbs total) (UPS GROUND -- FREE
Total: $1,585.71

Sparkfun.com

SparkFun micro:bit Breakout (with Headers) BOB-13989	40	$208.80

Amazon.com

TFD Supplies Wholesale Bulk Earbuds Headphones 100 Pack For Iphone, Android, MP3 Player - Black

13 of SE TL10 10-Piece Test Lead Set with Alligator Clips

5 of Bullet Face Copper Foil Tape with Double-sided Conductive (1/4inch X 21.8yards)- EMI Shielding,Stained Glass,Soldering,Electrical Repairs,Slug Repellent,Paper Circuits,Grounding (1/4inch)

DOSS Touch Wireless Bluetooth V4.0 Portable Speaker with HD Sound and Bass (Black)

3 of AmazonBasics 3.5 mm Coiled Stereo Audio Cable - 6.5 feet (2 Meters) Stretched Length

15 of AstroAI Digital Multimeter with Ohm Volt Amp and Diode Test

2 of 10K Ohm, 1/4 Watt, 5%, Carbon Film Resistors (pack of 100)
4 of TowerPro SG90 9G Micro Servo 5pk For RC Airplane Boat Helicopter Robot Controls

2 of Hakko CHP CSP-30-1 Wire Stripper, 30-20 Gauge Maximum Cutting Capacity

Sourced in South Africa

- AAA Batteries
- 220V soldering irons and solder

- 220V glue guns and glue sticks
- Tempera paint and some brushes
- Cardboard, plastic bottles
- Box cutters
- Masking and duct tape

Gotchas:

- The Microbit seems to choke when you use more than four or five GPIO pins – beware of overly complex GPIO projects
- Work out beforehand where students can save their work and find it again – individual folders, network shares, etc.
- Test everything beforehand and check current draws – the 1.5V buzzers I ordered didn't work except when batteries were brand new – not enough current
- AAA batteries (required for the Microbit) were not easy to find in South Africa – don't assume availability
- If you're working in a developing country or remote area, plan for bad internet access, or none – come prepared to work offline
- If possible, train some assistants beforehand
- Microbit pin voltage is 3.3V – there are lots of 5V sensors for Arduino that won't work
- Check the voltage of LEDs before ordering – I ordered a pack of five colors and only two of them (red and yellow) would light up at 3.3V
- There's a mostly undocumented trick to get many 5V PIR sensors to work at 3.3V – see this: http://techgurka.blogspot.com/2013/05/cheap-pyroelectric-infrared-pir-motion.html