Digital Watch

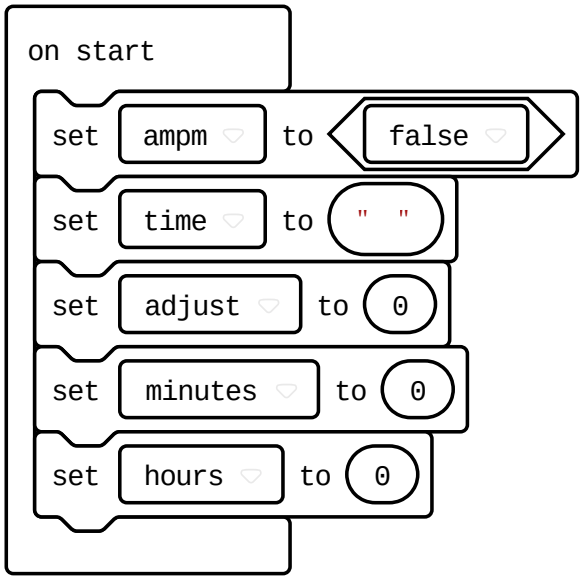
Put the power of time into your watch. Let's code up a real digital watch for your micro:bit!

Duration: ~20 minutes

Make the time variables

We need to make some variables to keep track of the time and for a few other things.

1. Go into **Basic** in the toolbox and pull an `on start` on to the workspace.
2. Ok, in **Variables** click on `Make a Variable`. Name the variable as `hours`. Drag out a `set to` block and change the name with the dropdown to `hours`. Place the variable into the `on start` block.
3. Repeat this 4 more times to make variables named `minutes`, `time`, `adjust`, and `ampm`.
4. Now, for the `set to` block for `time`, go to **Text** and drag a `" "` in and replace the `0`.
5. For the `ampm` variable, change the `0` there to a `false` from the **Logic** category.

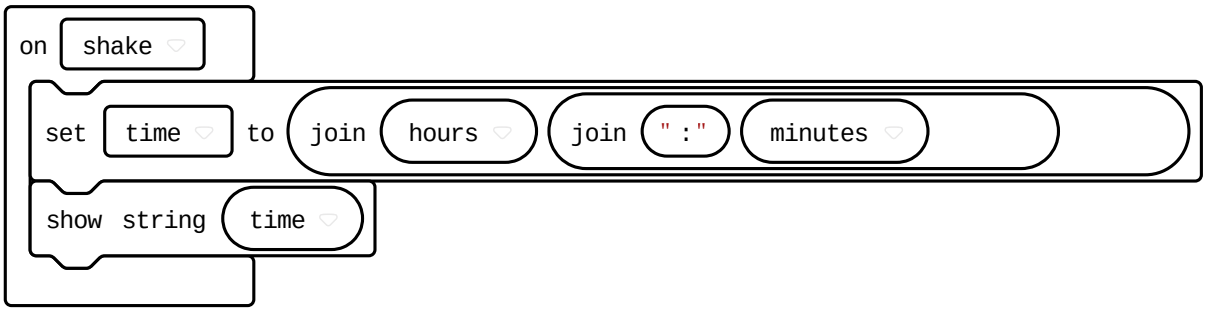



```
let hours = 0
let minutes = 0
let adjust = 0
let time = ""
let ampm = false
```

Display the time, kind of

So, let's try showing the time on the display. We aren't keeping time yet but we'll just see if we can make our watch show something.

1. Get in the **Input** category and pull out an `on shake`. We'll have our watch show the time when it's shaken.
2. Get another `set to` and put it into the `on shake`. Change the name to `time`.
3. Replace the `0` with a `join` from **Text**. Get another `join` and put it into the second slot of the first `join` you pulled out.
4. Change the `" "` in the first `join` to the `hours` variable. Change the text in the first slot of the second `join` to `":"`. And, change the last slot in the second `join` to the `minutes` variable.
5. Finally, stick in a `show string` below the `set to`. Switch the text inside to the variable `time`.
6. Download the code to you micro:bit and give it a shake. Did you see the time of "0:0" go by on the LEDs?



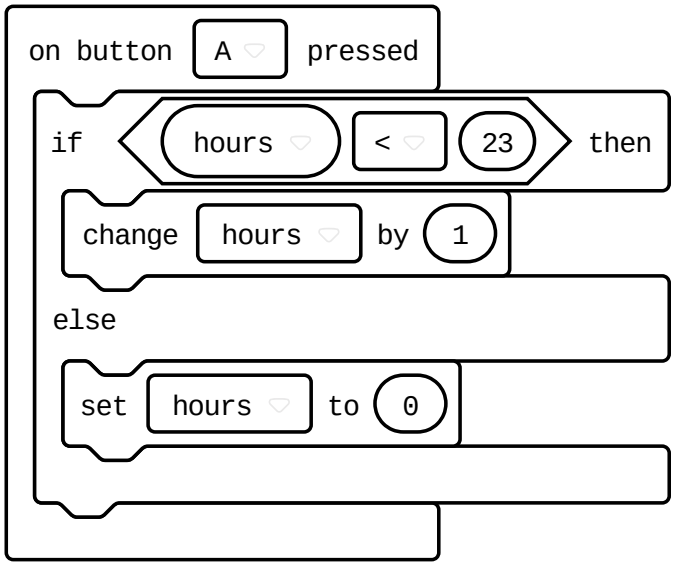
Set the time with buttons

There has to be a way to set the time on your watch. We'll use the buttons to set the current time. One button is for setting the hours and another button is for the minutes.

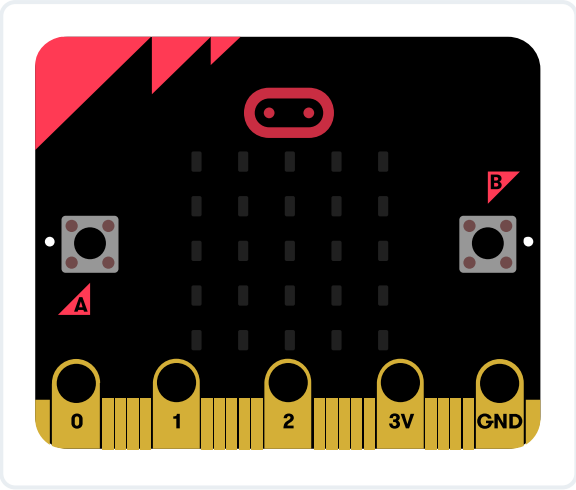
Set the hours

Let's make a way to set the hours for the watch.

1. In **Input**, find an `on button pressed` an put it somewhere on the workspace.
2. Get an `if then else` block from **Logic** and put it in the `on button pressed`.
3. From the same **Logic** category, get a `0 < 0` and replace the `false` condition with it.
4. Change the left `0` in the condition to the `hours` variable. Change `0` on the right to `23`. This limits our hour count to 23 hours.
5. In the `then` section, put a `change by` there. Select the `hours` variable name from the dropdown.
6. In the `else` section, put a `set to` there. Select the `hours` variable name from the dropdown and leave the `0`.



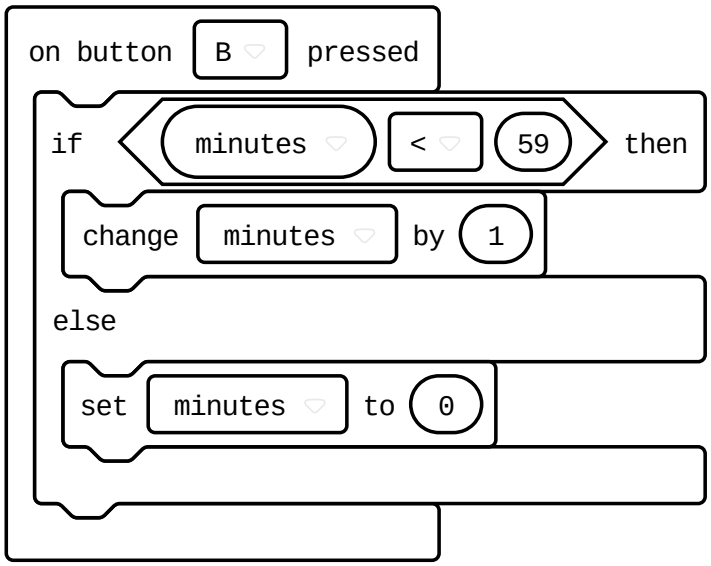

```
let hours = 0;
input.onButtonPressed(Button.A, () => {
    if (hours < 23) {
        hours += 1;
    } else {
        hours = 0;
    }
})
```



Set the minutes

Setting minutes is almost the same as setting hours but with just a few changes.

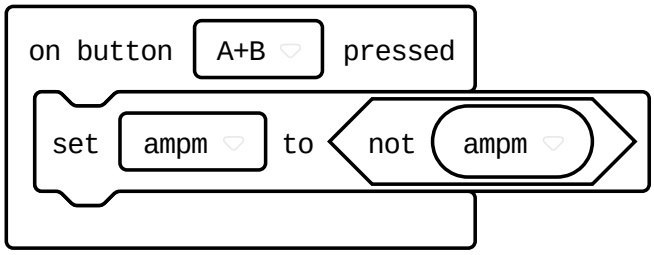
1. To make things easy, right click on `on button pressed` block and select the **Duplicate** option in the menu. This makes a copy of the original block.
2. In the new `on button pressed`, change the button to `B`.
3. Change every variable name from `hours` to `minutes`. Change the `23` in the `if` condition to `59`. This is the limit of minutes we count.



Select 24 hour or 12 hour time

Time is shown in either 24 hour or 12 hour format. We'll use one more button to choose which format to show. Using the 12 hour format adds an 'AM' or 'PM' at the end.

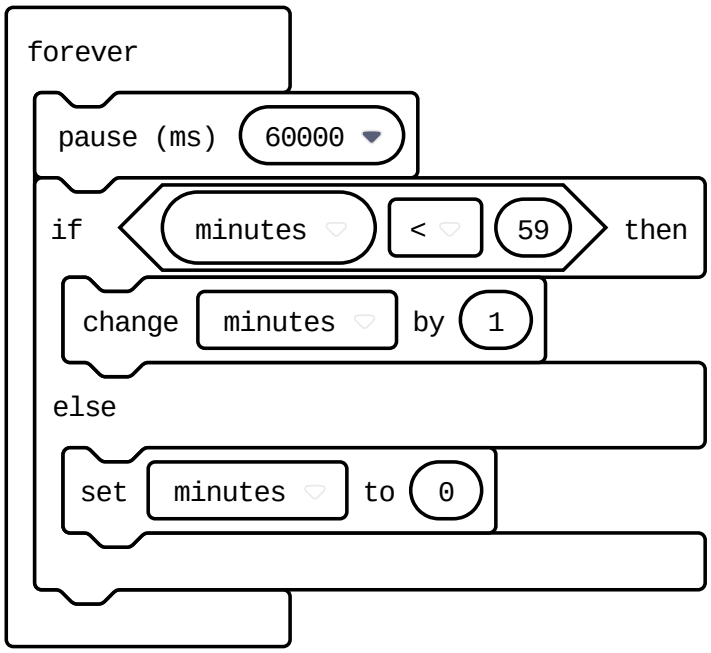
1. In **Input**, get an `on button pressed` an put it on the workspace. Change the button to `A+B`.
2. Grab a `set to`, put it in the block and change the variable to `ampm`. Put a `not` from **Logic** in where the `0` is.
3. Pick up a `ampm` from **Variables** and connect it on the right of the `not`. This switches our 24 hour format to 12 hour and back.



Make the timer tick

A watch really has three parts: the display, settings, and timer. We need a way to make the minutes and hours count up at the right time. Let's code the timer.

1. In **Basic**, get a `forever` loop out to the workspace.
2. Also in **Basic**, take out a `pause` an put it into the loop. Change the time from `100` to `60000`. The time is in milliseconds so we want to count each minute every 60000 milliseconds.
3. Below the `pause`, put a `if then else` block. Change the condition in the `if` to use a `0 < 0`.
4. Replace the `0` on the left with the `minutes` variable. Change the `0` on the right to `59`.
5. Put a `change by` into the `then`. Change the variable to `minutes`.
6. Get a `set to` and put it in the `else`. Again, change the variable to `minutes`.

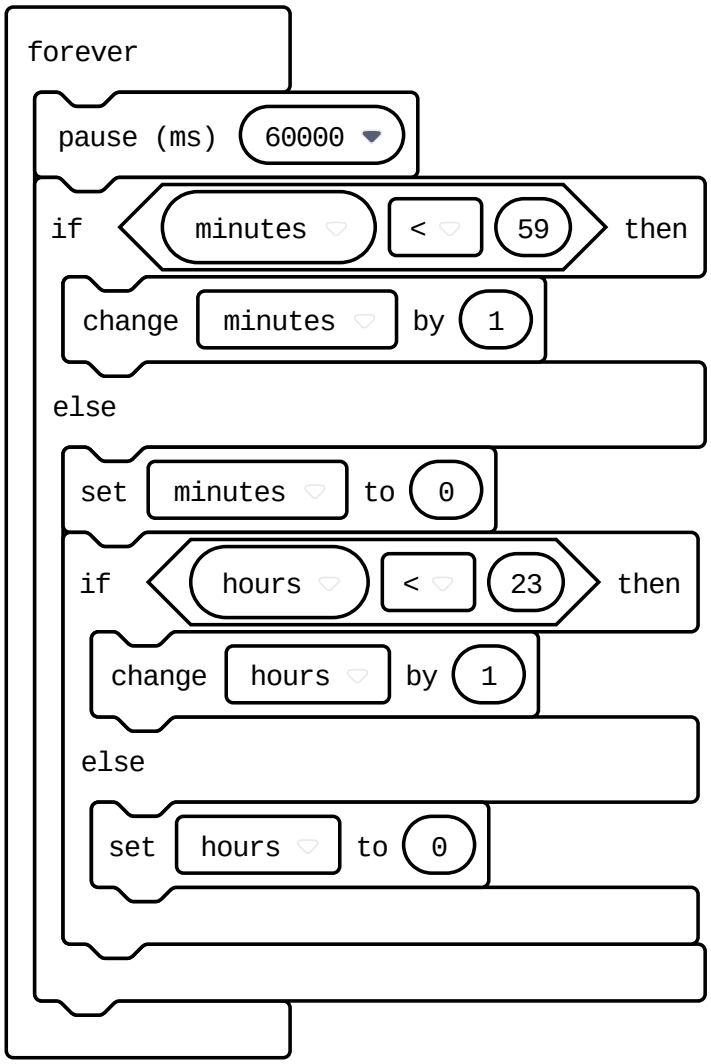


Keep on coding…

1. Now, take another `if then else` and put it just below the `set to` inside the first `else`.
2. In the second `if`, put in a `0 < 0` as the condition. Replace the left `0` with the `hours` variable. Change the right `0` to `23`. We count hours up to 23 until we go

back to 0 (midnight).

3. Put a `change by` into the second `then`. Change the variable to `hours`.
4. Get a `set to` and put it in the second `else`. Again, change the variable to `hours`. Ok, the timer's ready to tick.

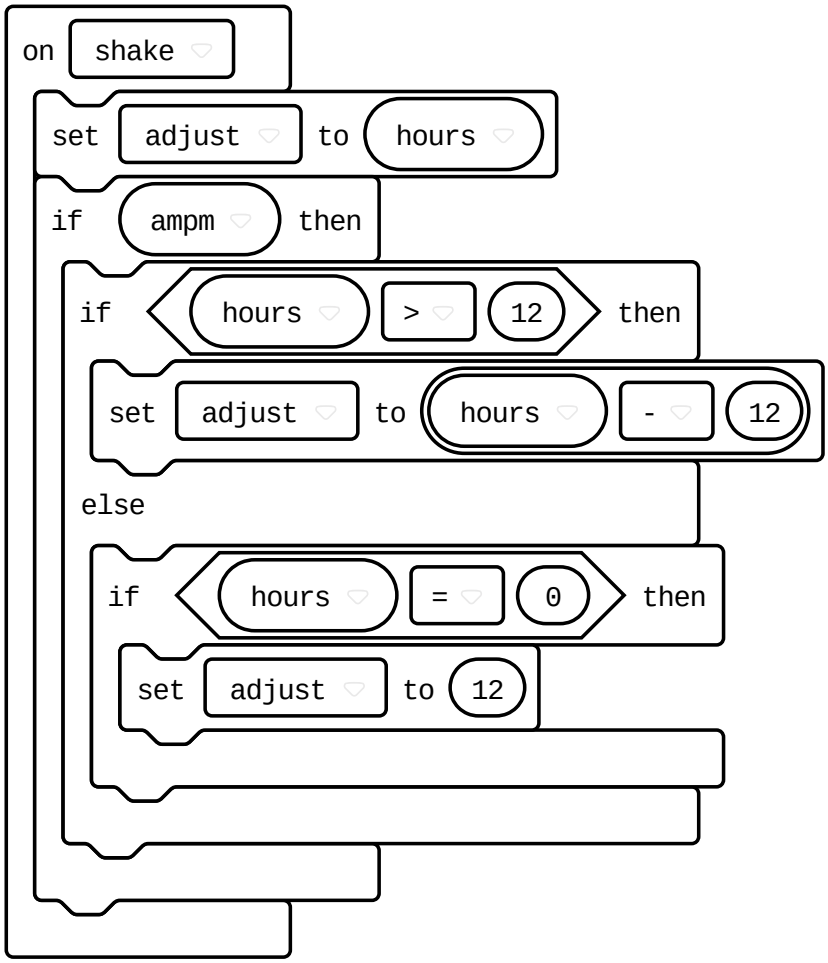


Shake and show…the time!

We're going back to the display code we made earlier. We'll now make it show the real time! This step is going to be busy but we'll get it done.

First, we have to code an adjustment for the hours number when we're using the 12 hour format.

1. Find the `on shake` block we coded earlier. Pull out and drag to the trash the blocks inside. We're starting fresh.
2. Pull out a `set to` an put it inside the `on shake`. Change the variable to `adjust`. Change the `0` on the right to the `hours` variable.
3. Get a `if then` and put it under the `set to`. Replace the condition with the `ampm` variable.
4. Grab a `if then else` and put it in the `then` part of the first `if then`. Change the condition to `0 < 0`. Replace the `0` on the left with the `hours` variable. Change the `0` on the right to `12`. Switch the `<` to a `>`.
5. Go get another `set to` and put it in the `then` of the second `if then else`. Change the variable to `adjust`. In **Math** take a `0 - 0` and replace the `0` in the `set to`. Change the `0` on the left to the `hours` variable and the `0` on the right to `12`.
6. Take one more `if then` and put it in the `else`. Change its condition to `0 = 0`. Put the `hours` variable in place of the `0` on the left.
7. Inside this last `if then` place a `set to`. Change the variable name to `adjust` and set the value to `12`.



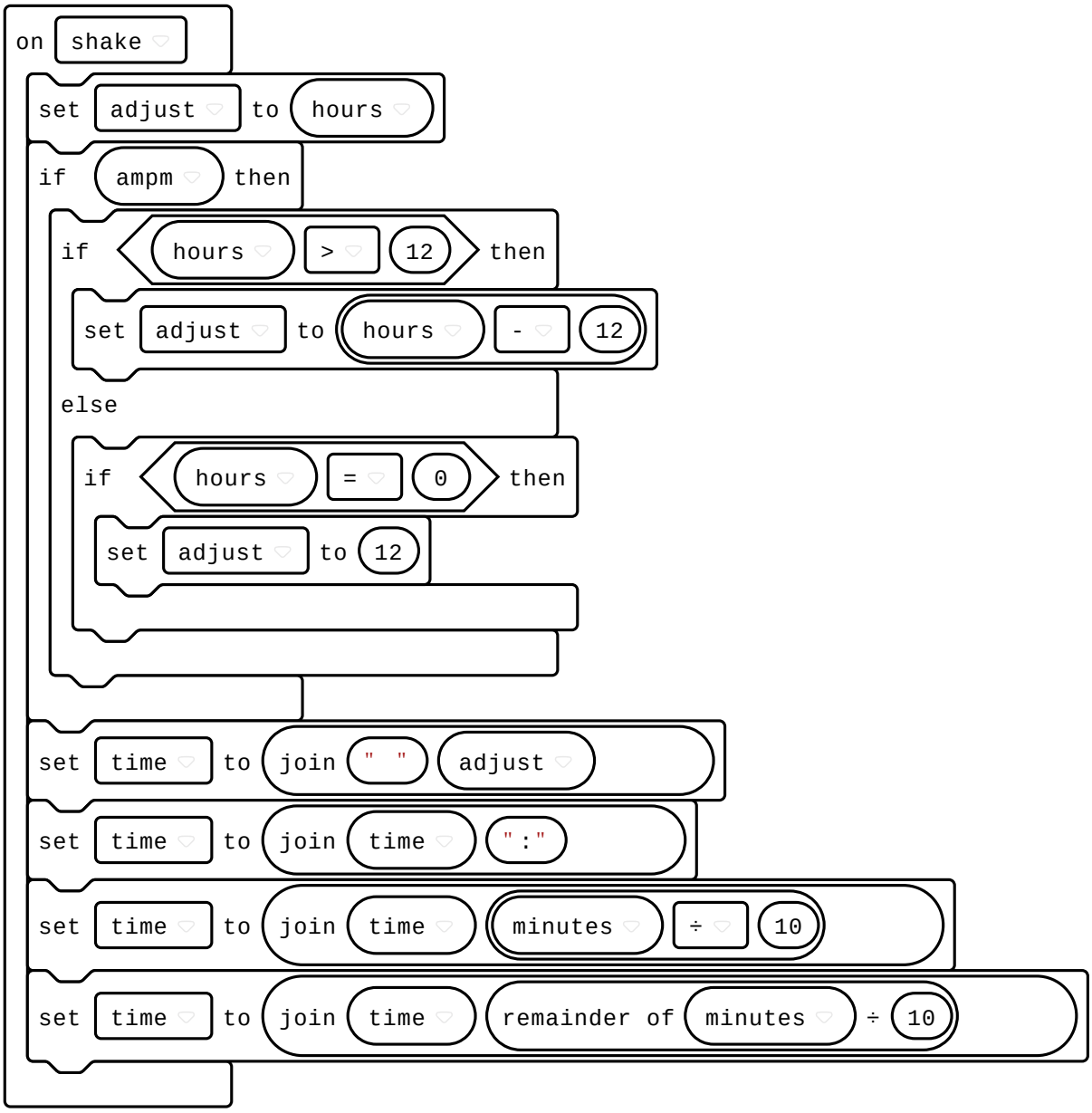
Keep on coding…

Now, we have to join up the hours and minutes to make text that will display on the watch.

1. At the bottom of the `on shake`, insert a `set to`. Change the variable name to `time`. Connect it to a `join` from **Text**.
2. Make 3 copies of this last `set to` using the **Duplicate** option in the menu when you right click on the block. Put the copies underneath each other so that all 4 are stacked together.
3. In the first `set to`, replace the second `""` in the `join` with the `adjust` variable.
4. With the second copy, change the first `""` in the `join` to the variable `time`. Change the second string in the `join` to `":"`.
5. In the third copy, change the first `""` in the `join` to the variable `time`. Change the second string in the `join` to division operator from **Math**. Set the left `0` to the

`minutes` variable and the right `0` to `10`.

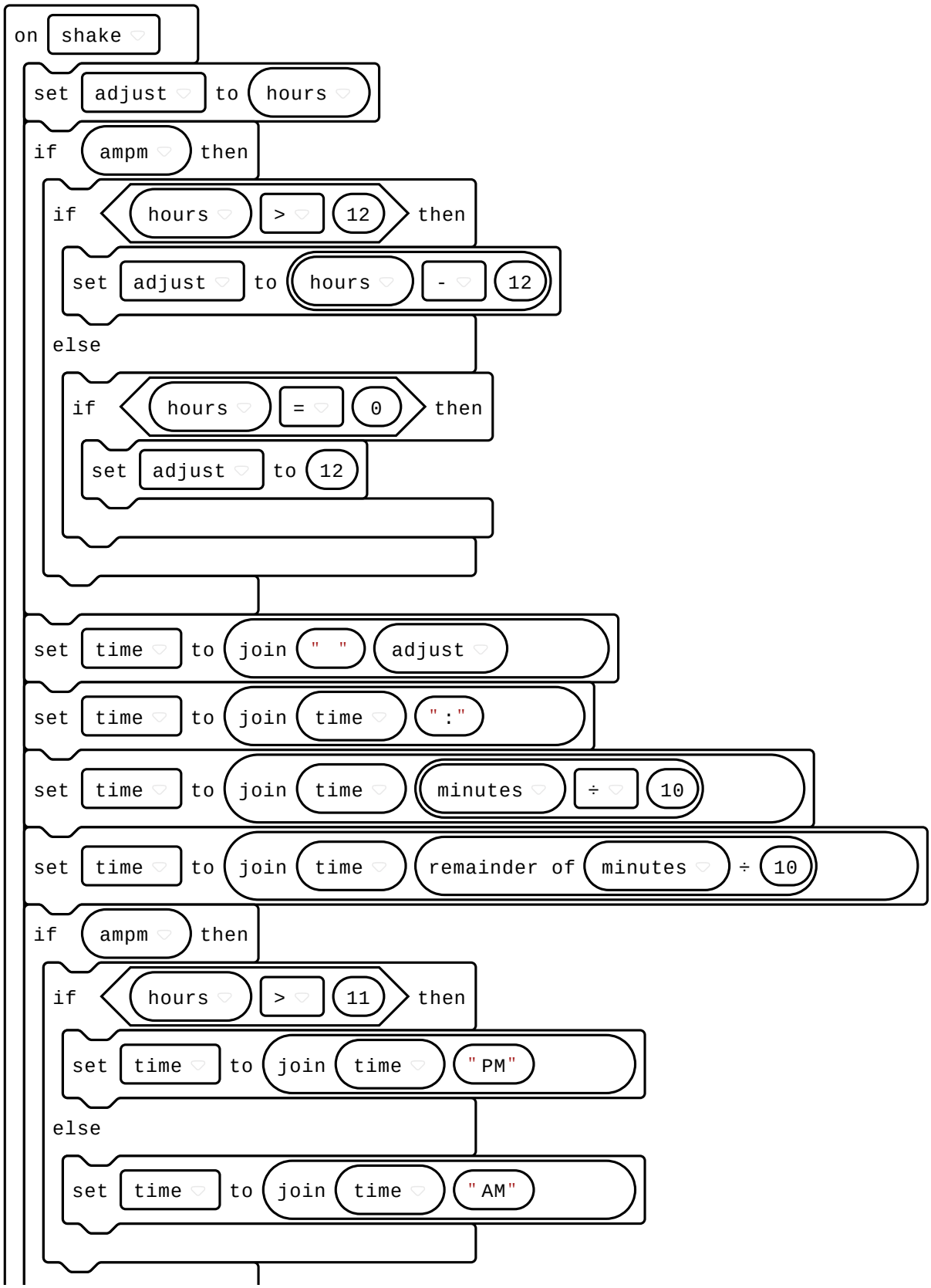
6. In the fourth copy, change the first `""` in the `join` to the variable `time`.
   Change the second string in the `join` to a `remainder of` in **Math**. Set the left `0` to the `minutes` variable and the right `0` to `10`.

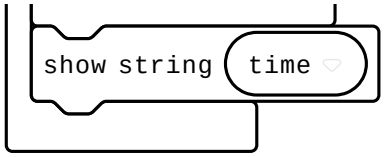


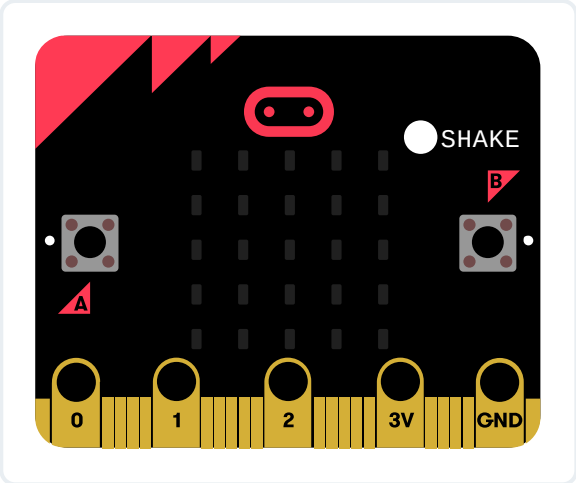
Keep on coding…

Ok, we're getting close to finishing now. Here we need to add the 'AM' or 'PM' if we are in 12 hour format. Then, finally, display the complete time string.

1. Put an `if then` block at the end of the `on shake`. Replace the `true` condition with the variable `ampm`.
2. Insert a `if then else` into this `if then`. Use a `0 < 0` as the condition. Change the left `0` to the `hours` variable. Change the right `0` to `11`. Switch the `<` to a `>`.
3. Place a `set to` in the `then`. Change the variable to `time` and attach a `join`. Make the first part of the `join` be the variable `time` and the second part to the text `"PM"`.
4. Do the exact same thing as in the last step but put the `set to` block in the `else` underneath. Make the second part of the `join` be `"AM"` this time.
5. Finally, at the very bottom of `on shake`, go get a `show string` from **Basic** and put it there. Change the string `"Hello!"` to the `time` variable.


on shake
set adjust to hours
if ampm then
if hours > 12 then
set adjust to hours - 12
else
if hours = 0 then
set adjust to 12
set time to join " " adjust
set time to join time ":"
set time to join time minutes ÷ 10
set time to join time remainder of minutes ÷ 10
if ampm then
if hours > 11 then
set time to join time "PM"
else
set time to join time "AM"





Complete!

Wow, so awesome! You've got your watch coded and ready to try. Go press the Download button and put your code on the micro:bit. When you shake it, it shows the current time.

Right now, it's showing 24 hour format: hours go from `0` to `23` and back to `0`. Press the **A+B** buttons together to change to 12 hour format: hours go from `12` to `12` with `1` through `11` in between. It has either `"AM"` or `"PM"` at the end.

To set it to the current time, you use the **A** and **B** buttons. The **A** button moves the current hour up by one each time it's pressed. The **B** button moves the minutes up by one every time it's pressed.

Now that you can tell time on your micro:bit who knows what you will accomplish next. Only, time will tell!