

The Booklet

Projects with Micro: bit

- WEB AL-Jazri2
- Micro:bit.....4
- Digital Watch6

WEB AL-Jazri

Objective:

Students will design and program a digital watch using the micro:bit, exploring the evolution of timekeeping technology. This project will introduce fundamental programming skills and concepts, empowering students to assemble and code their own functional smart clock. By drawing on historical advancements in clockmaking, such as the contributions of Al-Jazri, the project encourages students to explore the intersection of ancient engineering innovations and modern technological developments. Through this process, students will gain a deeper understanding of how past inventions continue to influence contemporary digital technologies.

Learnable Skills:

- **Block-Based Programming:** Writing scripts using MakeCode's block interface to control and display time on the micro:bit.
- **Button Integration:** Programming the micro:bit's buttons to allow for setting and adjusting the time dynamically.
- **LED Display Utilization:** Displaying time effectively using the micro:bit's 5x5 LED matrix.
- **Timekeeping Logic:** Implementing timers, counters, and logic to accurately track hours, minutes, and seconds.
- **Hardware Assembly:** Connecting and powering the micro:bit to create a standalone, functional smart clock.
- **Creative Customization:** Designing and adding personalized features to the clock, such as alarms, animations, or custom time formats.
- **Debugging and Optimization:** Identifying and resolving programming challenges to enhance functionality and performance.
- **Historical Engineering Inspiration:** Drawing inspiration from Al-Jazri's historical innovations to connect modern timekeeping technologies with ancient engineering.
- **Introduction to AI Concepts:** Understanding how the micro:bit processes inputs and makes decisions based on programmed logic, introducing foundational concepts of artificial intelligence.

Hands-on Practice Tools:

- micro:bit.
- block coding.

Learnable Skills and Discoverable Tools:

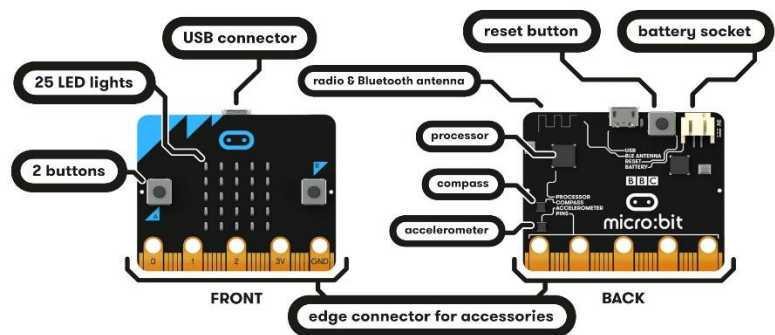
- **Block-Based Programming.**
- **Teamwork.**
- **Design Thinking.**
- **Debugging and Problem-Solving.**

Micro: bit

1 - What is Micro: bit? :

is a small, programmable computer designed to help students learn about technology, coding, and electronics.

2 - Micro: Bit Components:



Front Side:

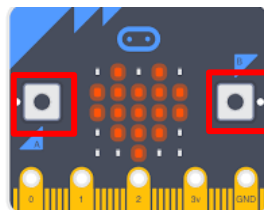
- LED Display (5x5 Grid):

A grid of 25 small LED lights that can show numbers, letters, shapes, and even simple animations. You can program it to display images, text as shown below:



- Buttons (A and B):

Two programmable buttons (A and B) that can be used for interacting with the micro: bit, such as controlling games or triggering actions in your program.



- **USB Connector:**
A micro-USB port that is used to power the micro: bit and to transfer programs (code) from your computer to the micro: bit.
- **Edge Connector for Accessories:**
The golden strip at the bottom has 25 pins. These can be connected to external components like sensors, motors, or LEDs, making the micro: bit highly customizable for projects.

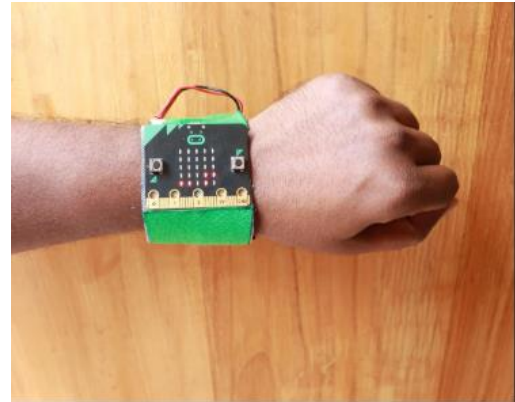
Back Side:

- **Radio & Bluetooth Antenna:**
Allows the micro: bit to wirelessly communicate with other devices, including other micro: bits or smartphones, using radio signals or Bluetooth.
- **Reset Button:**
A small button to restart the micro: bit. This is useful if you want to run your program from the beginning or troubleshoot.
- **Battery Socket:**
A slot for attaching an external battery pack to power the micro: bit when not connected to a computer.
- **Processor:**
The (brain) of the micro: bit that runs the code you upload. It processes all inputs and controls the outputs.
- **Compass:**
A sensor that detects the Earth's magnetic field, allowing the micro: bit to function as a digital compass or detect nearby magnets.
- **Accelerometer:**
A sensor that detects motion and orientation. It can tell if the micro: bit is being shaken, tilted, or moved, which is great for motion-sensitive projects.

Digital Watch

Overview

The **Digital Watch Project** transforms the Micro:bit into a functional timekeeping device using block-based coding. This project supports **Time Management Systems** and introduces learners to the basics of **Event-Driven Programming** and **Logical Sequencing**. It's an engaging way to develop problem-solving skills, enhance coding creativity, and build a strong foundation in microcontroller applications.



Goals Addressed :

- **Goal 1. Logical Systems**
Apply variables, loops, and conditional statements to manage timekeeping functions, including setting hours, minutes, and alarms.
- **Goal 2. Sensor Integration**
Utilize micro:bit's button inputs to adjust the time, switch between modes (clock, stopwatch, alarm), and trigger specific watch functions.
- **Goal 3. Product Prototyping**
Design, build, and test a fully operational digital watch prototype with real-time display and functional alarms.
- **Goal 4. Pitching Skills**
Effectively present and demonstrate the digital watch, explaining its features, coding structure, and real-world applications.

Why It's Fun

- **Interactive Gameplay:** Players can interact with the micro:bit to view time in real-time and challenge each other in timed activities, making it a fun, hands-on experience.
- **Randomized Outcomes:** The digital watch allows for timed alarms or reminders, adding an element of suspense as players never know when the timer will alert them.
- **Creative Coding:** Customize the watch's display with different designs, colors, or special animations, allowing for a personalized touch and a creative coding challenge for users.

Digital Watch

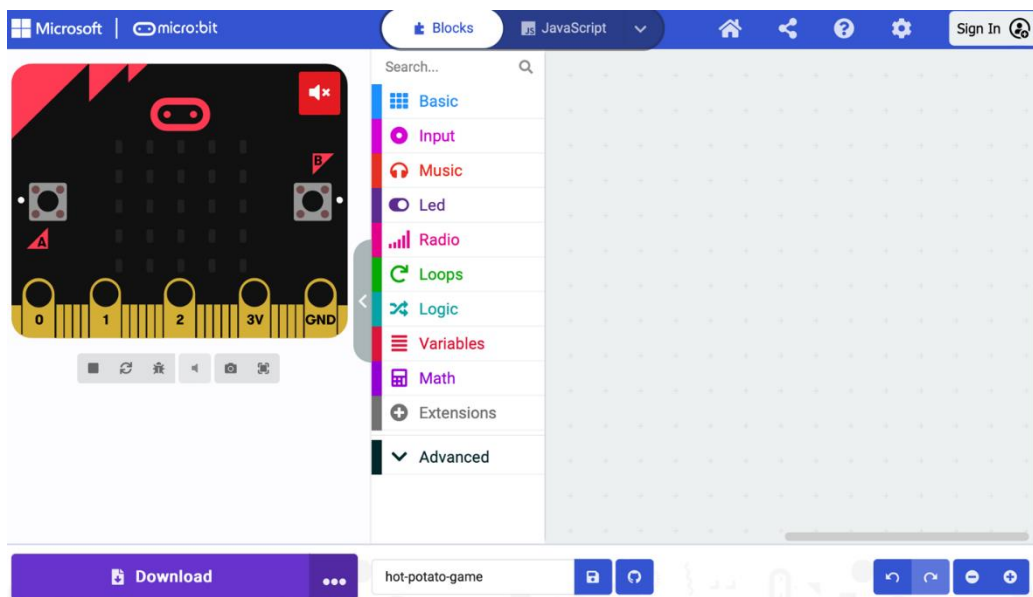
This Guide on How to Create the Digital Watch with Micro:bit

The Project Idea

This project turns the micro:bit into a **Digital Watch**, allowing users to keep track of time using a simple, but functional design.

To implement this project, we will use the **MakeCode Editor by Microsoft**.

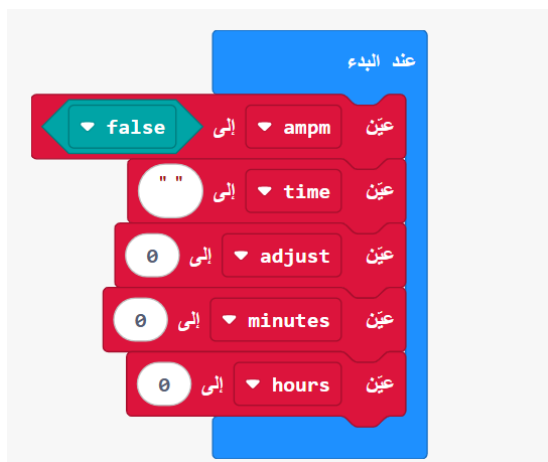
- **Step 1:**
Search for MakeCode Editor in google:
<https://makecode.microbit.org/#editor>
- **Step 2:**
Click on new project and name it: Digital watch
Then this screen will appear



○ Step 3:

Make the time variables:

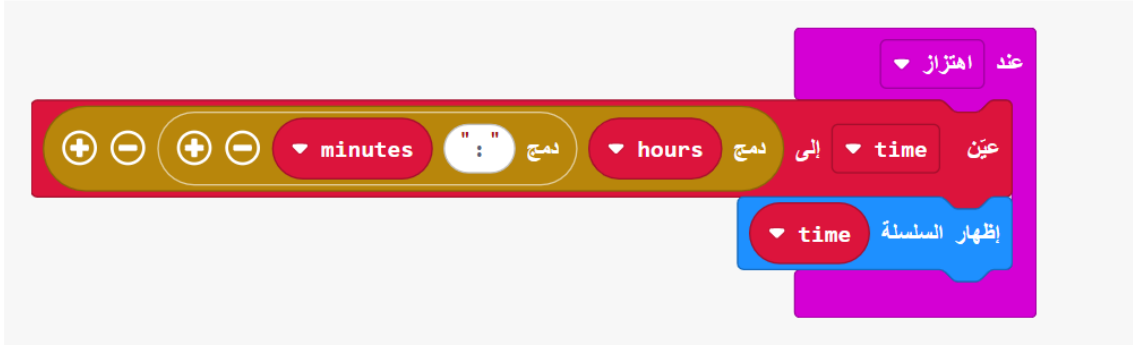
1. Go into **Basic** in the toolbox and pull an **on start** on to the workspace.
2. Ok, in **Variables** click on **Make a Variable**. Name the variable as `hours`. Drag out a **set to** block and change the name with the dropdown to `hours`. Place the variable into the **on start** block.
3. Repeat 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.



○ Step 4:

Display the time:

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 **دمج** from **Text**. Get another **دمج** and put it into the second slot of the first **دمج** you pulled out.
4. Change the " " in the first **دمج** to the `hours` variable. Change the text in the first slot of the second **دمج** to " : ". And, change the last slot in the second **دمج** 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?

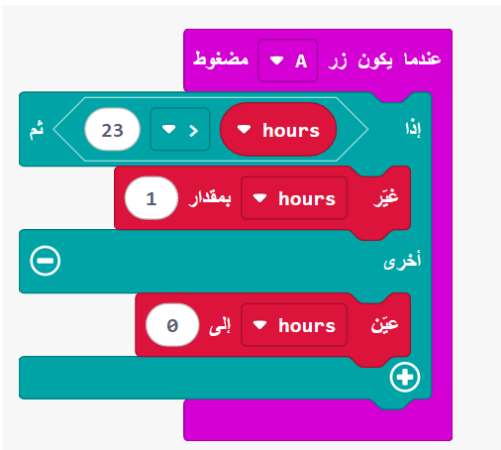


○ **Step 5:**

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.

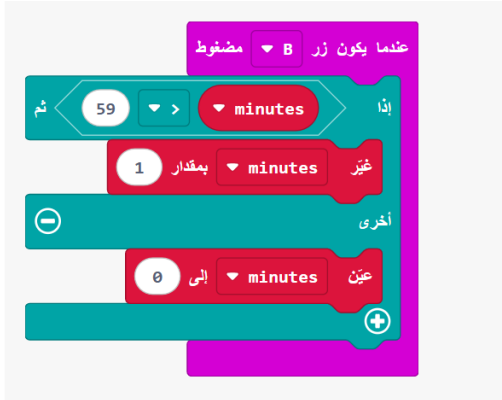
Set the hours:

1. In **Input**, find an **on button pressed** and 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 `true` 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.



Set the minutes:

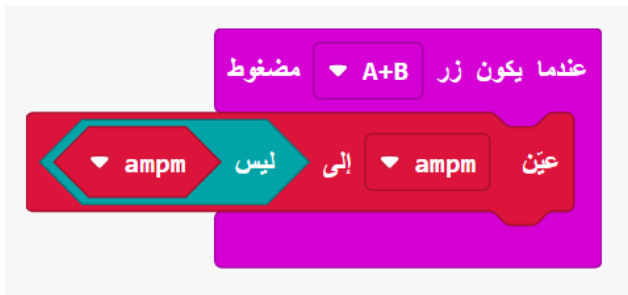
1. To make things easy, right click on **on button pressed** block and select the **Duplicate** option in the menu. This makes a copy of the original block.
2. In the new **on button pressed**, change the button to B.
3. Change every variable name from `hours` to `minutes`. Change the 23 in the **if**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



○ Step 6:

Select 24 hour or 12 hour time:

1. In **Input**, get an **on button pressed** and 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.



○ Step 7:

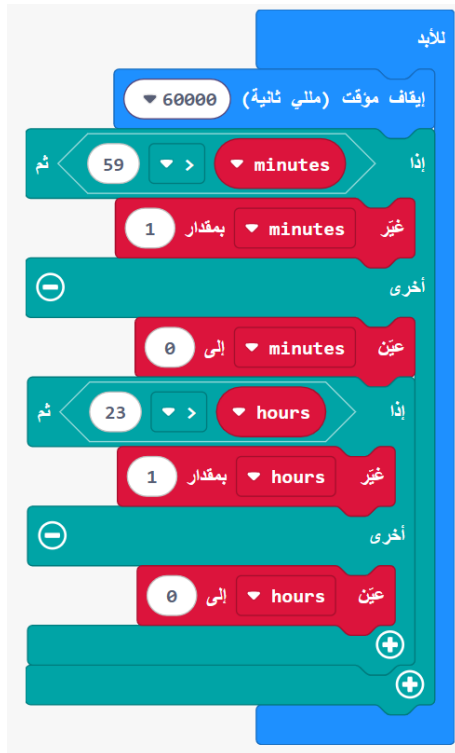
Make the timer tick: A watch really has three parts: the display, settings, and timer. We need a way to make the minutes and hours count up at the right time. Let's code the timer.

1. In **Basic**, get a **forever** loop out to the workspace.
2. Also in **Basic**, take out a **pause** and 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...

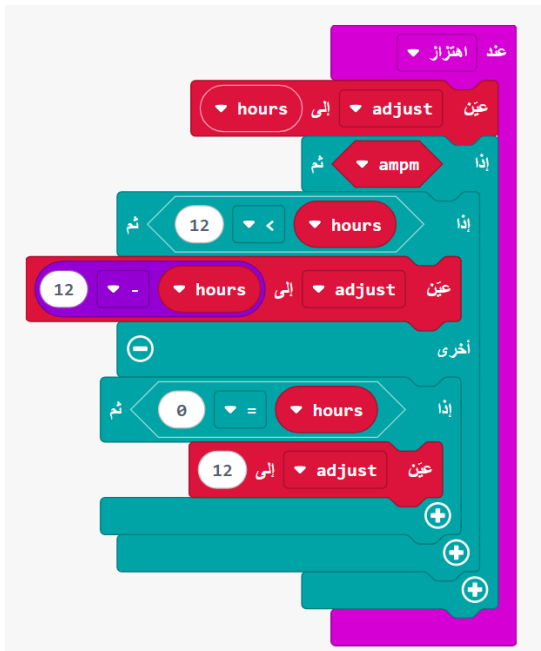
1. Now, take another **if then else** and put it just below the **set to** inside the first **else**.
2. In the second **if**, put in a $0 < 0$ as the condition. Replace the left 0 with the `hours` variable. Change the right 0 to 23. We count hours up to 23 until we go back to 0 (midnight).
3. Put a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



○ **Step 8:**

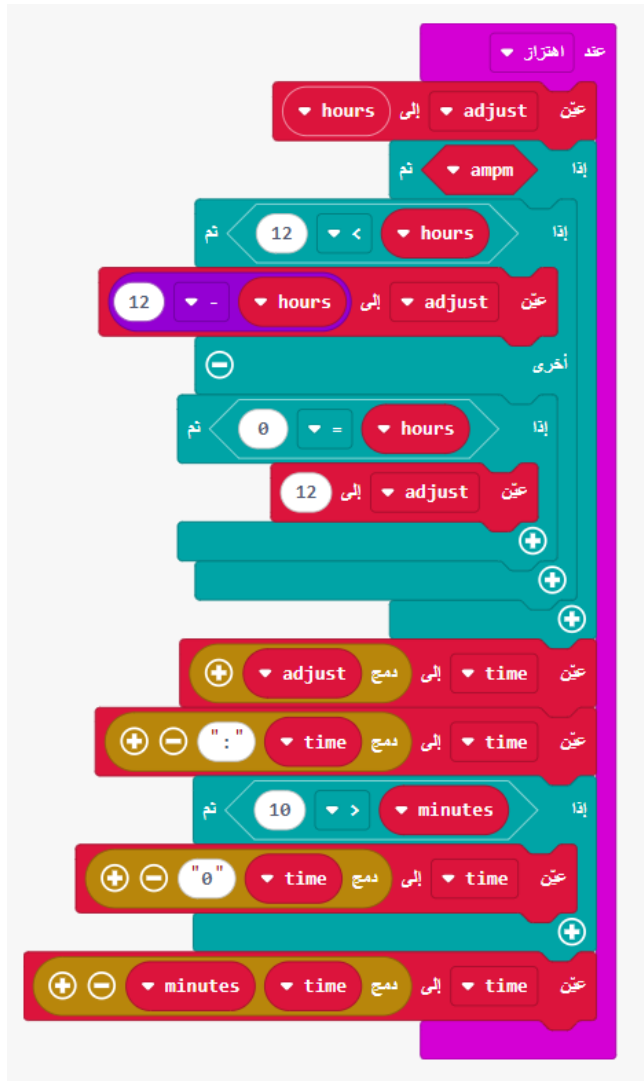
Shake and show...the time!: We're going back to the display code we made earlier. We'll now make it show the real time! This step is going to be busy but we'll get it done. First, we have to code an adjustment for the hours number when we're using the 12 hour format.

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** and 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.



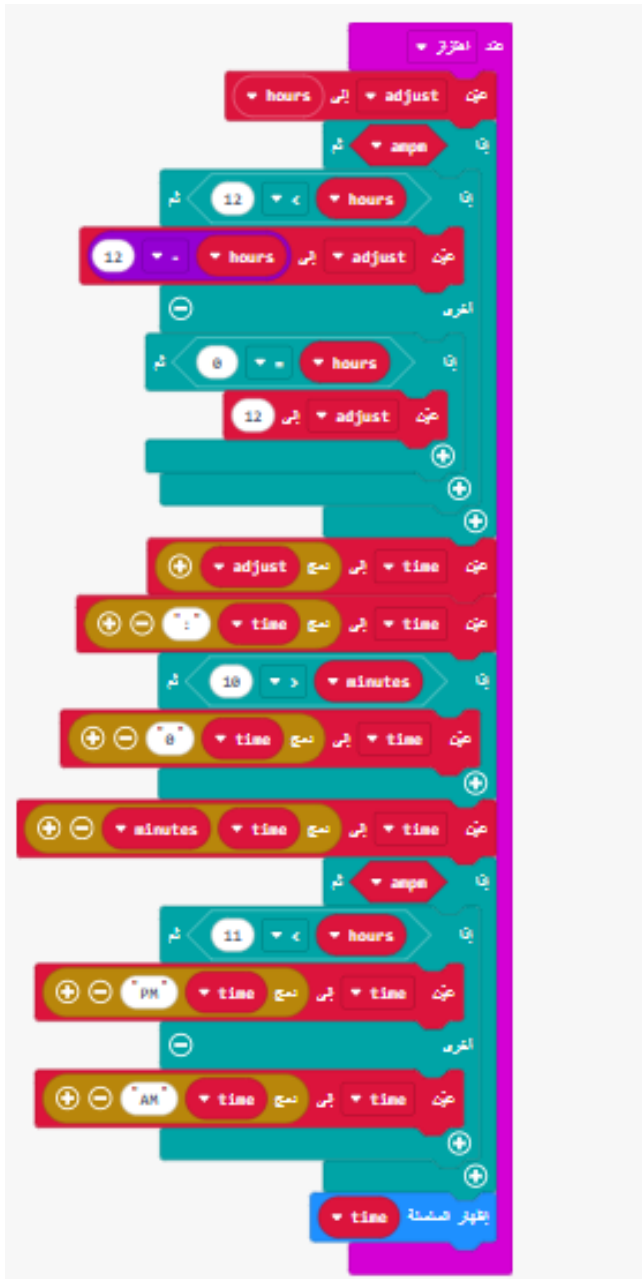
Keep on coding...

1. At the bottom of the **on shake**, insert a **set to**. Change the variable name to time. Connect it to a **دمج** 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 **دمج** with the `adjust` variable.
4. With the second copy, change the first "" in the **دمج** to the variable `time`. Change the second string in the **دمج** to " : ".
5. For the third copy, change the first "" in the **دمج** to the variable `time`. Change the second string in the **دمج** to "0".
6. Surround that third copy with an **if then** block. Put a `0 < 0` in as the condition for the **if then**. Pull a `minutes` variable in where the first 0 is. Change the second 0 to be a 9.
7. In the fourth copy, change the first "" in the **دمج** to the variable `time`. Change the second string in the **دمج** to a `minutes`.



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.

8. Put an **if then** block at the end of the **on shake**. Replace the `true` condition with the variable `ampm`.
9. 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 `>`.
10. Place a **set to** in the **then**. Change the variable to `time` and attach a **دمج**. Make the first part of the **دمج** be the variable `time` and the second part to the text `"PM"`.
11. 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 **دمج** be `"AM"` this time.
12. 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.



By diving into the **Digital Watch** project, you're transforming basic coding into a practical, interactive experience that helps you understand the logic behind timekeeping. This activity blends creativity, problem-solving, and hands-on coding, empowering you to build a functional and customizable digital watch. Be proud—you're turning programming into a tool for real-world applications and innovation into something you can wear on your wrist! 🕒 🎨