ARDUINO MODULE ATTINY85

The **ATtiny85** is a small, low-power 8-bit microcontroller from Microchip (formerly Atmel). It is part of the AVR family and is commonly used in projects where space and power consumption are critical. The ATtiny85 can be programmed using the Arduino IDE, making it a popular choice for simple projects.

Here's a guide to using the **ATtiny85 with Arduino**:

---

1. Setting Up the Arduino IDE for ATtiny85

To program the ATtiny85 using the Arduino IDE, you need to add support for the ATtiny series.

1. **Install the Arduino IDE**: Download and install the latest version of the Arduino IDE from arduino.cc.
2. **Add ATtiny Core**:
   - Open the Arduino IDE.
   - Go to **File > Preferences**.
   - In the "Additional Boards Manager URLs" field, add the following URL:

     Copy

     ```
     https://raw.githubusercontent.com/damellis/attiny/ide-1.6.x-boards-manager/package_damellis_attiny_index.json
     ```
   - Click **OK**.
3. **Install the ATtiny Board Package**:
   - Go to **Tools > Board > Boards Manager**.
   - Search for "attiny" and install the **ATtiny** package by David A. Mellis.
4. **Select the ATtiny85 Board**:
   - Go to **Tools > Board** and select **ATtiny25/45/85**.
   - Set the following options:
     - **Processor**: ATtiny85
     - **Clock**: Internal 8 MHz (or your desired clock speed)

ARDUINO MODULE ATTINY85

- **Programmer**: Arduino as ISP (or your preferred programmer)

---

2. Wiring the ATtiny85

To program the ATtiny85, you'll need an Arduino (e.g., Arduino Uno) as an ISP (In-System Programmer).

1. **Connect the Arduino Uno to the ATtiny85**:
   - Arduino Uno (ISP) -> ATtiny85
   - 5V -> VCC
   - GND -> GND
   - Pin 10 (SS) -> RESET (Pin 1)
   - Pin 11 (MOSI) -> MOSI (Pin 5)
   - Pin 12 (MISO) -> MISO (Pin 6)
   - Pin 13 (SCK) -> SCK (Pin 7)
2. **Add a 10µF Capacitor**:
   - Place a 10µF capacitor between the **RESET** and **GND** pins on the Arduino Uno to prevent it from resetting during programming.

---

3. Uploading a Program to the ATtiny85

1. **Open the Arduino IDE** and write your code (e.g., a simple blink program).
2. **Select the Programmer**:
   - Go to **Tools > Programmer** and select **Arduino as ISP**.
3. **Burn the Bootloader** (optional but recommended):
   - Go to **Tools > Burn Bootloader**. This sets the correct fuses for the ATtiny85.
4. **Upload the Sketch**:
   - Click **Sketch > Upload Using Programmer** to upload your code to the ATtiny85.

ARDUINO MODULE ATTINY85

4. Example Code

Here's a simple blink example for the ATtiny85:

cpp
Copy

```cpp
void setup() {
  pinMode(0, OUTPUT); // Pin 0 on ATtiny85 (physical pin 5)
}

void loop() {
  digitalWrite(0, HIGH); // Turn on LED
  delay(500);            // Wait 500ms
  digitalWrite(0, LOW);  // Turn off LED
  delay(500);            // Wait 500ms
}
```

---

5. Using the ATtiny85 Standalone

Once programmed, you can use the ATtiny85 in your project without the Arduino Uno. Just connect:

- **VCC** to 5V (or 3.3V if using a lower voltage).
- **GND** to ground.
- Add any necessary components (e.g., resistors, LEDs, sensors).

---

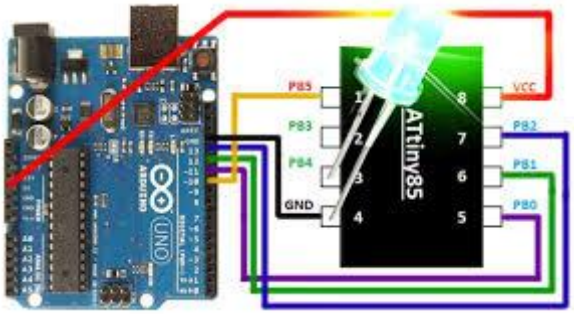
6. Pinout of ATtiny85

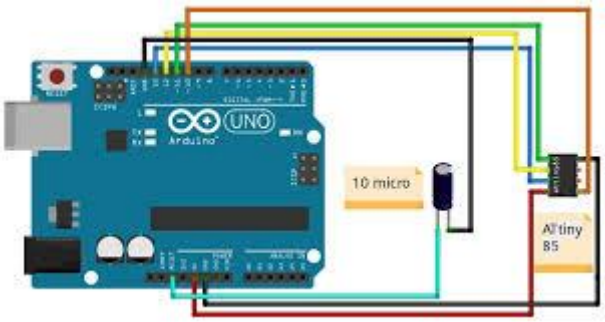
Here's the pinout for reference:

Copy

```
      +-----+
 RESET |1   8| VCC
  PB3  |2   7| PB2 (SCK)
  PB4  |3   6| PB1 (MISO)
  GND  |4   5| PB0 (MOSI)
      +-----+
```

ARDUINO MODULE ATTINY85





7. Tips

- The ATtiny85 has limited memory (8KB flash, 512B RAM), so keep your code small.
- Use the **Arduino as ISP** or a dedicated programmer like the USBasp.
- For more advanced projects, consider using libraries optimized for the ATtiny85.