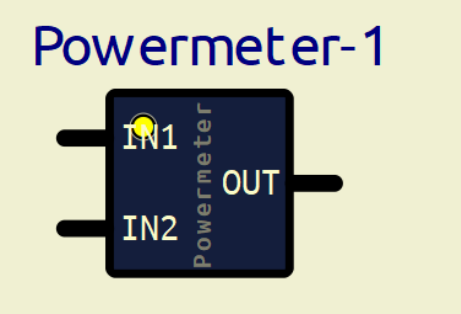
SimulIDE

Adding a Scripted Component – DC Power Meter

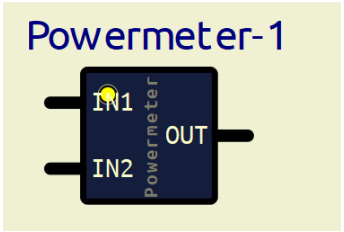


🧩 The Purpose of `.as`, `.mcu`, and `.package` Files

File Extension	Purpose	Contains
`.as`	**Scripted behavior**	The actual **logic/instructions** that run inside the component. Think of it like firmware for a microcontroller or internal code for a logic chip.
`.mcu`	**Configuration and port mapping**	The **component definition file**: sets the name, links to the `.as` file, defines ports and clock speed.
`.package`	**Pin layout mapping** *(optional/advanced)*	Defines the **physical pin layout**: maps port bits (like `RA0`, `RB1`) to specific pin numbers and names for UI purposes.

Three Files:

1. Powermeter.package
2. Powermeter.mcu
3. Powermeter.as



Powermeter.package | Powermeter.mcu | Powermeter.as

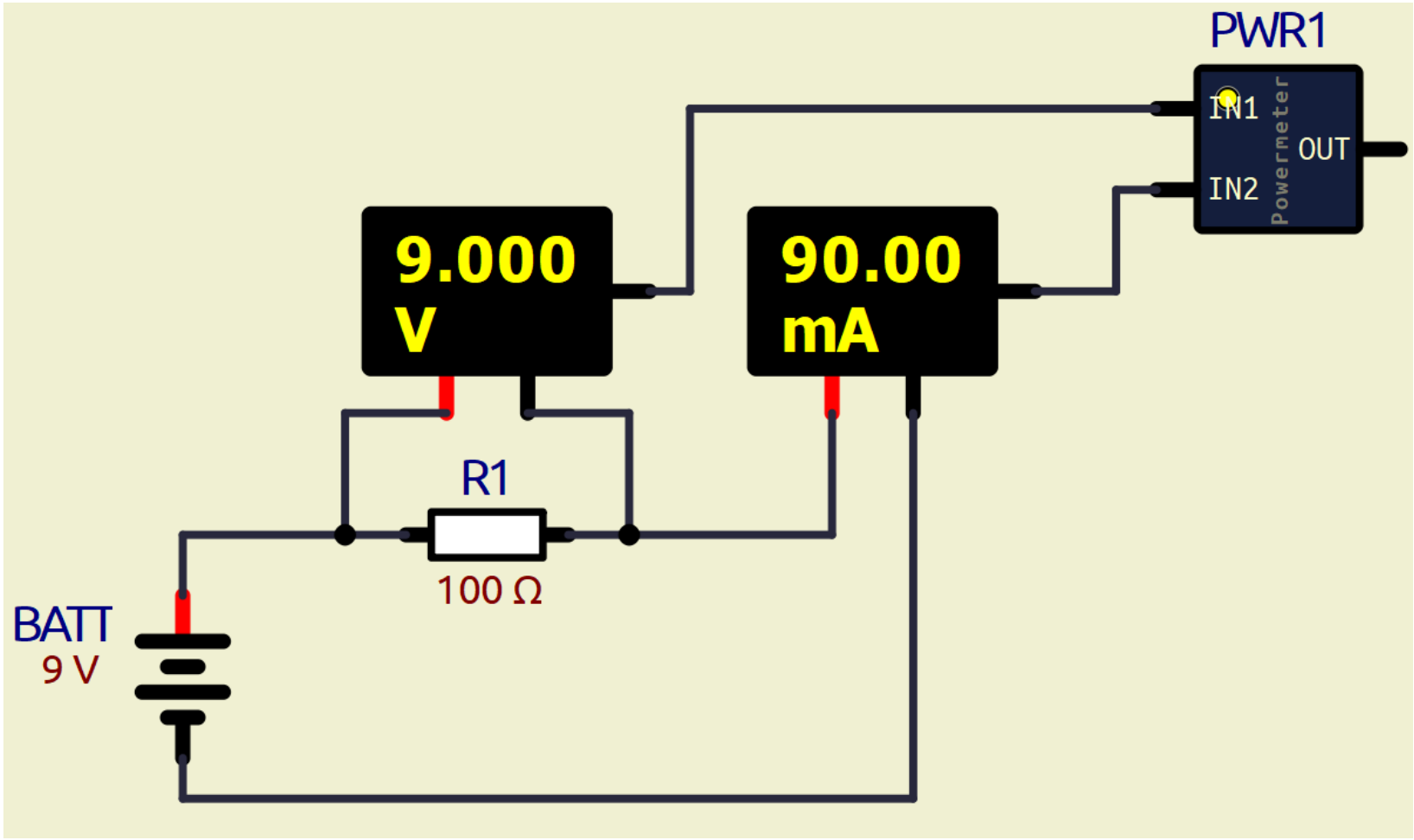
```
<!DOCTYPE SimulIDE>

<packageB name="Powermeter" width="4" height="4" background="" type="Scripted">

    <pin type=""    xpos="-8"   ypos="8"    angle="180" length="8"  space="0"   id="IN1"    label="IN1" />
    <pin type=""    xpos="-8"   ypos="24"   angle="180" length="8"  space="0"   id="IN2"    label="IN2" />
    <pin type=""    xpos="40"   ypos="16"   angle="0"   length="8"  space="0"   id="OUT"    label="OUT" />

    <!-- Link to MCU definition -->
    <mcuFile>Powermeter.mcu</mcuFile>

</packageB>
```

Powermeter.package | Powermeter.mcu | Powermeter.as

```
<!DOCTYPE SimulIDE>

<iou name="Powermeter" core="scripted" script="Powermeter.as" clock="1000000">

    <ioport name="PORTA" pins="IN1,IN2" />

    <ioport name="PORTB" pins="OUT" />

</iou>
```

Powermeter.package | Powermeter.mcu | Powermeter.as

```
IoPin@ inputPin1  = component.getPin("IN1");
IoPin@ inputPin2  = component.getPin("IN2");
IoPin@ outputPin  = component.getPin("OUT");

double power = 0;

void setup()
{
    print("Power setup() Doing Nothing");
}

void reset()
{
    print("Power reset()");

    inputPin1.setPinMode(1);
    inputPin2.setPinMode(1);
    outputPin.setPinMode(3);
    outputPin.setVoltage(0);

    inputPin1.changeCallBack(element, true);
    inputPin2.changeCallBack(element, true);
}

void voltChanged()
{
    double input1 = inputPin1.getVoltage();
    double input2 = inputPin2.getVoltage();
    power = input1 * input2;

    outputPin.setVoltage(power);
    print("Power = " + power + " W");
}
```



Simulation Time:
00:00:44 s 600 ms 000 µs 000 ns 001 ps

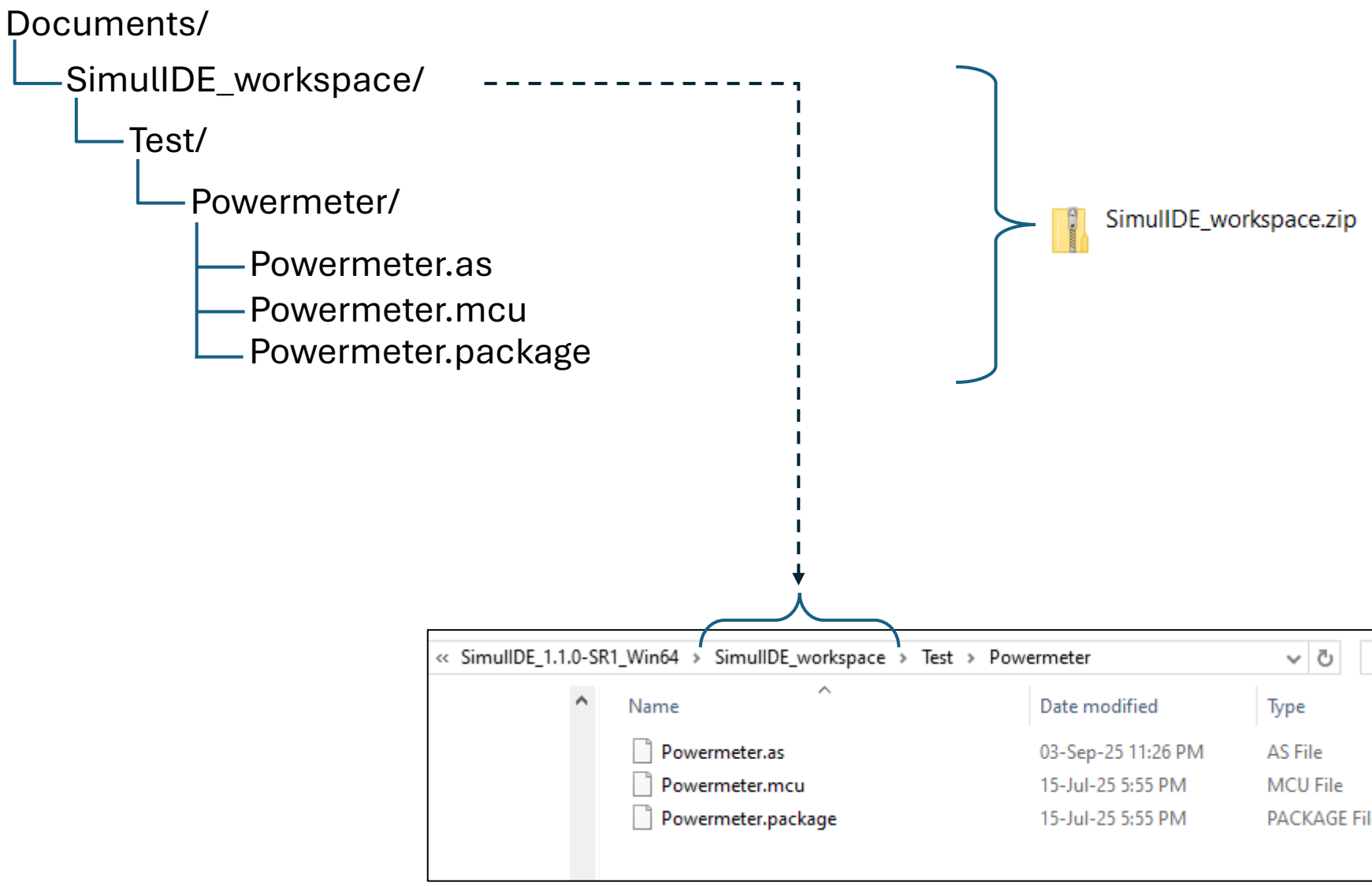
Target Speed:	100.00 %
Real Speed:	000.00 %
Engine Load:	000.00 %
Update Load:	000.00 %
Real FPS:	0
Main MCU:	PWR1

```
Power reset()
    Simulation Running...

Power = 0.809984 W

    Simulation Stopped

------------------------------------------------
```

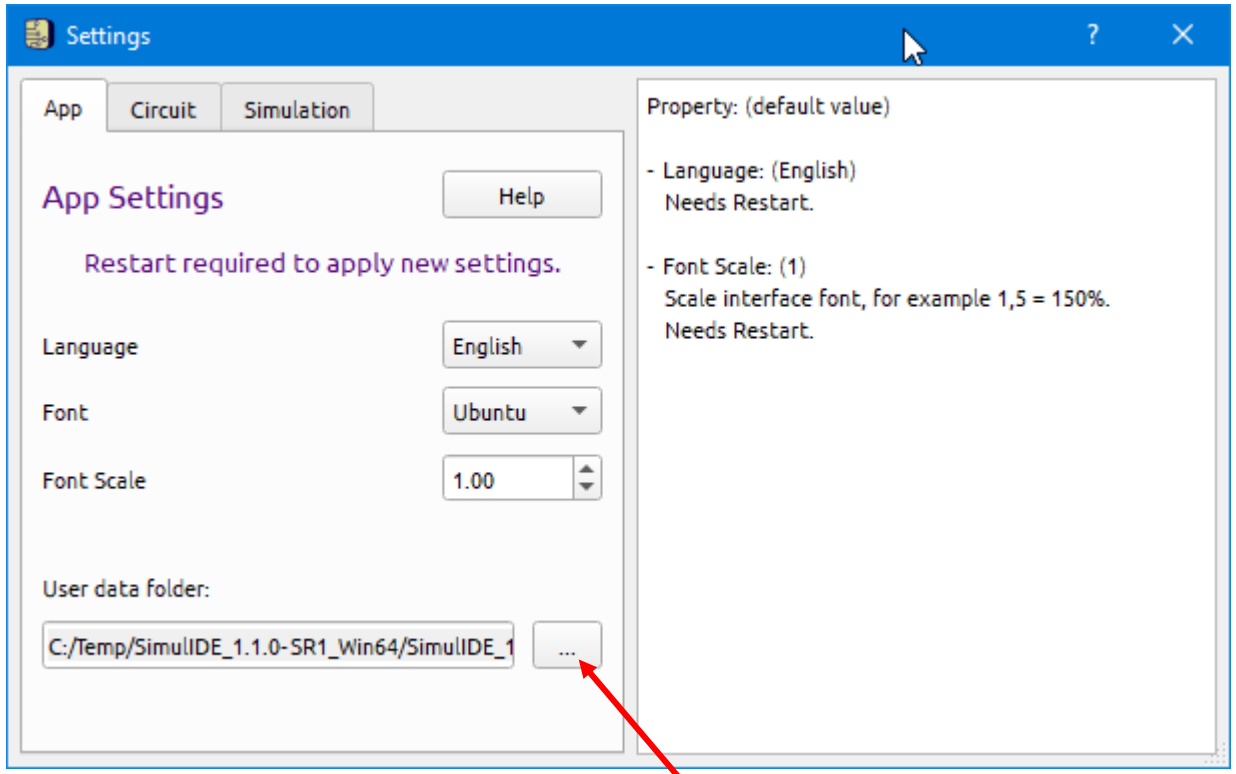
$P = 9V \times 90 \times 10^{-3} A$

$= 0.81W$

Typical Directory Structure on Windows:



Setup User Folder



C:/Temp/SimulIDE_1.1.0-SR1_Win64/SimulIDE_1.1.0-SR1_Win64/SimulIDE_workspace/

Requires a restart to recognize **new scripted components**

THE END

Powermeter.package | Powermeter.mcu | Powermeter.as

```
<!DOCTYPE SimulIDE>

<packageB name="Powermeter" width="4" height="4" background="" type="Scripted">

    <pin type=""    xpos="-8"   ypos="8"    angle="180" length="8"  space="0"   id="IN1"    label="IN1" />
    <pin type=""    xpos="-8"   ypos="24"   angle="180" length="8"  space="0"   id="IN2"    label="IN2" />
    <pin type=""    xpos="40"   ypos="16"   angle="0"   length="8"  space="0"   id="OUT"    label="OUT" />

    <!-- Link to MCU definition -->
    <mcuFile>Powermeter.mcu</mcuFile>

</packageB>

```

Powermeter.package | Powermeter.mcu | Powermeter.as

```
<!DOCTYPE SimulIDE>

<iou name="Powermeter" core="scripted" script="Powermeter.as" clock="1000000">

    <ioport name="PORTA" pins="IN1,IN2" />

    <ioport name="PORTB" pins="OUT" />

</iou>
```

Powermeter.package | Powermeter.mcu | Powermeter.as

```
IoPin@ inputPin1  = component.getPin("IN1");
IoPin@ inputPin2  = component.getPin("IN2");
IoPin@ outputPin  = component.getPin("OUT");

double power = 0;

void setup()
{
    print("Power setup() Doing Nothing");
}

void reset()
{
    print("Power reset()");

    inputPin1.setPinMode(1);
    inputPin2.setPinMode(1);
    outputPin.setPinMode(3);
    outputPin.setVoltage(0);

    inputPin1.changeCallBack(element, true);
    inputPin2.changeCallBack(element, true);
}

void voltChanged()
{
    double input1 = inputPin1.getVoltage();
    double input2 = inputPin2.getVoltage();
    power = input1 * input2;

    outputPin.setVoltage(power);
    print("Power = " + power + " W");
}
```