

MakerEdu I2C Dual DC Motor Driver - User Manual

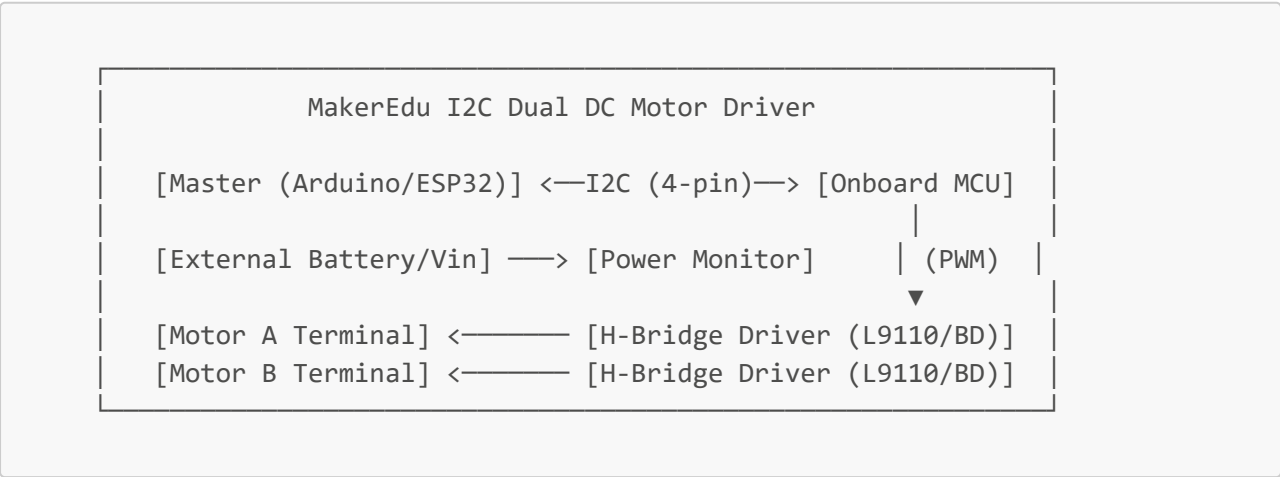
Document Version: 1.1.0

Target Hardware Models:

- **MKE-M17:** MakerEdu I2C Dual DC Motor Driver L9110 (Silkscreen: **I2C DRIVER L9110**)
 - **MKE-M18:** MakerEdu I2C Dual DC Motor Driver BD62130 (Silkscreen: **I2C DRIVER BD62130**)
- Compatible Controllers:** Arduino (Uno, Mega, Nano), ESP32, ESP8266, **BBC micro:bit (MakeCode Extension)**, Raspberry Pi, STM32

1. Product Overview

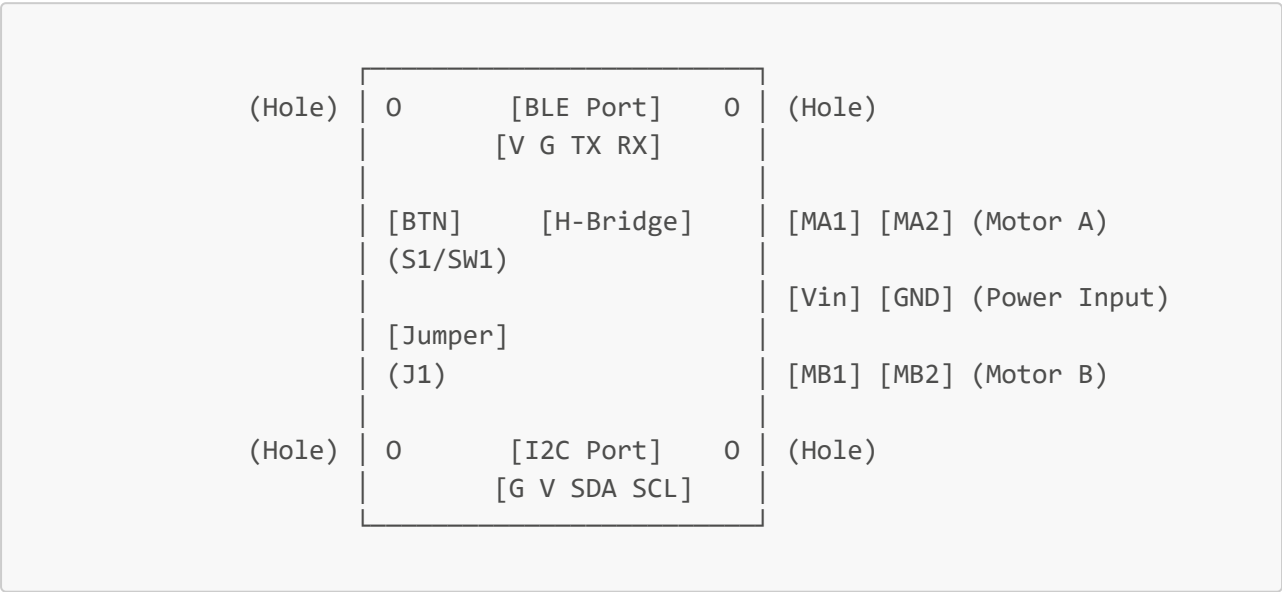
The **MakerEdu I2C Dual DC Motor Driver** is a smart, high-efficiency motor control module designed for robotics, smart vehicles, and automation projects. Powered by an onboard 32-bit ARM Cortex-M0+ microcontroller, it offloads PWM generation, direction switching, timing management, and voltage monitoring from the Master controller over a simple **2-wire I2C interface**.



Hardware Specifications: MKE-M17 (L9110) vs. MKE-M18 (BD62130)

Feature	MKE-M17 (L9110)	MKE-M18 (BD62130)
Product SKU	MKE-M17	MKE-M18
Silkscreen Marking	I2C DRIVER L9110	I2C DRIVER BD62130
Logic Voltage (VCC)	3.3V - 5.0V DC	3.3V - 5.0V DC
Motor Supply Voltage (Vin)	6.0V - 9.0V DC (e.g. 2S Li-ion / 6x AA)	8.0V - 13.0V DC (e.g. 3S LiPo / 12V Lead-Acid)
Continuous Current / Channel	0.8 A	1.0 A
Peak Current / Channel	1.5 A	2.5 A (Built-in thermal & OCP protection)
PWM Frequency	500 Hz – 2000 Hz	500 Hz – 20000 Hz
Recommended Applications	2WD/4WD Mini Toy Cars, TT Motors, N20 Motors	High-torque 370/520 DC Motors, Heavy Robotics, Mecanum Chassis

2. Hardware Interface & Pinout



Port & Terminal Descriptions

1. I2C Connector (4-Pin MakerEdu Standard):

- **GND**: Common ground.
- **VCC**: Logic power supply (3.3V or 5V from Master).
- **SDA**: I2C Data line (with onboard **10kΩ pull-up resistor**).
- **SCL**: I2C Clock line (with onboard **10kΩ pull-up resistor**).

2. BLE / UART Connector (4-Pin):

- **VCC, GND, TX, RX**: Dedicated interface for **MKE-M15 Bluetooth UART Module**.
- **Failsafe Feature**: When controlled via Dabble App over Bluetooth, the MCU continuously monitors heartbeat packets. If connection is lost or out of range, all motors immediately cut off PWM and come to a safe emergency stop.

3. Motor Output Terminals (Screw Terminals):

- **MA1, MA2**: Motor A output (includes direction indicator LEDs **MA1** & **MA2**).
- **MB1, MB2**: Motor B output (includes direction indicator LEDs **MB1** & **MB2**).

4. External Power Input Terminal:

- **Vin**: Motor positive power (+6V..9V for MKE-M17, +8V..13V for MKE-M18). Protected by SS34 reverse-polarity diode.
- **GND**: Power ground.

5. Power Selection Jumper / Switch (J1):

- **J1 = ON**: Enables the onboard 5V LDO regulator to power MCU and logic directly from **Vin**. (*⚠ Use ONLY when operating standalone without connecting VCC from I2C Master*).
- **J1 = OFF (Default)**: Disables onboard 5V LDO. MCU and logic take power from the I2C port (VCC pin from Master) or UART port.

(⚠ When using MKE-M15 Bluetooth module on the UART port, power **MUST** be supplied via the I2C Master VCC pin).

6. Multi-function Onboard Push Button (S1 / SW1):

- **Single Click:** Runs built-in motor self-test routine (spins Motor A CW/CCW, then Motor B CW/CCW).
- **Long Press for 4 Seconds:** Restores factory defaults and resets the I2C address back to 0x40 (Note: Hardware currently does not feature a dedicated LED indicator for this event; the address resets automatically after 4 seconds).

3. Wiring Diagram

Connecting to Arduino Uno (J1 = OFF)

Arduino Uno Pin	MakerEdu Motor Driver Pin	Description
5V	VCC	Logic Power
GND	GND	Common Ground
A4	SDA	I2C Data
A5	SCL	I2C Clock
External Battery (+)	Vin (Terminal)	Motor Power Supply (+6V..9V for M17, +8V..13V for M18)
External Battery (-)	GND (Terminal)	Motor Power Ground
DC Motor 1	MA1 / MA2	Motor A
DC Motor 2	MB1 / MB2	Motor B

4. Arduino Library Installation

You can install and use the driver library via either of the following two methods:

Method 1: Via the **MKE_ONE** Ecosystem Library (Recommended)

1. Open Arduino IDE, navigate to **Tools > Manage Libraries...** (**Ctrl + Shift + I**).
2. Search for **MKE_ONE** and click **Install**.
3. When prompted by Arduino IDE to install library dependencies, choose **"Install All"** (the IDE will automatically fetch **MakerEdu_I2C_MotorDriver**).
4. Access pre-configured examples directly from the Arduino menu:
 - **File > Examples > MKE_ONE > Module > MKE_M17_I2C_Motor_L9110** (for MKE-M17)
 - **File > Examples > MKE_ONE > Module > MKE_M18_I2C_Motor_BD62130** (for MKE-M18)

Method 2: Direct Driver Installation

1. Open Arduino IDE, navigate to **Tools > Manage Libraries....**
 2. Search for **MakerEdu I2C Motor Driver** and click **Install**.
-

5. C++ API Reference

Basic Code Template

```
#include <Wire.h>
#include <MKE_I2C_MotorDriver.h>

// You can declare using MKE_I2C_MotorDriver, MKE_M17_MotorDriver, or
// MKE_M18_MotorDriver
MKE_M17_MotorDriver motorDriver;

void setup() {
  Serial.begin(9600);
  Wire.begin();           // Master initializes I2C bus
  motorDriver.begin();    // Bind driver to address 0x40
}

void loop() {
  motorDriver.motorA_CW(200); // Motor A forward at speed 200/255
  delay(1000);
  motorDriver.stopAll();     // Stop all motors
  delay(1000);
}
```

Complete API Function Table

Function	Parameters	Return Type	Description
<code>begin(address, wire)</code>	<code>uint8_t address = 0x40, TwoWire &wire = Wire</code>	<code>void</code>	Configure I2C address & Wire instance (<i>Call <code>Wire.begin()</code> in <code>setup()</code> first</i>)
<code>motorA_CW(speed, duration_ms)</code>	<code>uint8_t speed (0-255), uint16_t duration_ms = 0</code>	<code>void</code>	Motor A Clockwise. <code>duration_ms = 0</code> runs continuously, <code>> 0</code> auto-stops
<code>motorA_CCW(speed, duration_ms)</code>	<code>uint8_t speed (0-255), uint16_t duration_ms = 0</code>	<code>void</code>	Motor A Counter-Clockwise
<code>stopMotorA()</code>	None	<code>void</code>	Stops Motor A immediately
<code>motorB_CW(speed, duration_ms)</code>	<code>uint8_t speed (0-255), uint16_t duration_ms = 0</code>	<code>void</code>	Motor B Clockwise
<code>motorB_CCW(speed, duration_ms)</code>	<code>uint8_t speed (0-255), uint16_t duration_ms = 0</code>	<code>void</code>	Motor B Counter-Clockwise
<code>stopMotorB()</code>	None	<code>void</code>	Stops Motor B immediately
<code>stopAll()</code>	None	<code>void</code>	Stops both Motor A and Motor B simultaneously
<code>getVin()</code>	None	<code>uint32_t</code>	Reads motor supply voltage (Vin) in mV (e.g. <code>8400</code> = 8.4V)
<code>setAddress(newAddress)</code>	<code>uint8_t newAddress (1-126)</code>	<code>void</code>	Sets new I2C address and saves to onboard EEPROM
<code>getAddress()</code>	None	<code>uint8_t</code>	Reads current I2C address stored in EEPROM
<code>getModuleId()</code>	None	<code>uint8_t</code>	Reads module type identifier (returns <code>3</code>)
<code>getFirmwareVersion()</code>	None	<code>uint32_t</code>	Reads firmware build date/version (e.g. <code>260331</code>)
<code>setPwmFrequency(freq_hz)</code>	<code>uint32_t freq_hz</code>	<code>void</code>	Sets PWM frequency (default: 500 Hz, recommended: 500–2000 Hz)

Function	Parameters	Return Type	Description
setPwmMA1(val), setPwmMA2(val), setPwmMB1(val), setPwmMB2(val)	uint8_t val (0-255)	void	Raw low-level PWM output to individual H-bridge gates

6. Practical Application Examples

Example 1: Basic Speed & Direction Ramping

```
#include <Wire.h>
#include <MKE_I2C_MotorDriver.h>

MKE_I2C_MotorDriver motorDriver;

void setup() {
  Serial.begin(9600);
  Wire.begin();
  motorDriver.begin();
}

void loop() {
  // Accelerate Motor A forward
  for (int spd = 0; spd <= 255; spd += 25) {
    motorDriver.motorA_CW(spd);
    delay(100);
  }
  delay(1000);
  motorDriver.stopMotorA();
  delay(500);

  // Reverse Motor A
  motorDriver.motorA_CCW(200);
  delay(1500);
  motorDriver.stopAll();
  delay(2000);
}
```

Example 2: Non-Blocking Timed Run & PID Watchdog Safety

```
#include <Wire.h>
#include <MKE_I2C_MotorDriver.h>

MKE_I2C_MotorDriver motorDriver;

void setup() {
  Serial.begin(9600);
  Wire.begin();
  motorDriver.begin();
}

void loop() {
  // Autonomous movement: Drive forward for 2.5 seconds and auto-stop
  motorDriver.motorA_CW(180, 2500); // speed=180, duration=2500ms
  motorDriver.motorB_CW(180, 2500);

  // Master is free to do sensor processing or calculations without blocking
  delay(4000);

  // PID Watchdog Loop: Stream commands every 100ms with 200ms timeout.
  // If Master crashes or communication is interrupted, motors safely halt
  within 200ms!
  for (int i = 0; i < 50; i++) {
    int speedA = 200;
    int speedB = 190;
    motorDriver.motorA_CW(speedA, 200);
    motorDriver.motorB_CW(speedB, 200);
    delay(100);
  }
  motorDriver.stopAll();
  delay(3000);
}
```

Example 3: Battery Monitoring & Low-Voltage Alarm

```
#include <Wire.h>
#include <MKE_I2C_MotorDriver.h>

MKE_I2C_MotorDriver motorDriver;

void setup() {
  Serial.begin(9600);
  Wire.begin();
  motorDriver.begin();
}

void loop() {
  uint32_t vin_mV = motorDriver.getVin();
  float vin_V = vin_mV / 1000.0;

  Serial.print(F("Battery Voltage: "));
  Serial.print(vin_V, 2);
  Serial.println(F(" V"));

  // Check 2S Li-ion threshold (cutoff at 6.6V)
  if (vin_mV > 0 && vin_mV < 6600) {
    Serial.println(F(">>> WARNING: Low Battery! Stopping motors to protect cells. <<<"));
    motorDriver.stopAll();
    while (true) { delay(1000); } // Halt execution
  }

  delay(1000);
}
```

Example 4: Cascading Multiple Drivers on One I2C Bus

```
#include <Wire.h>
#include <MKE_I2C_MotorDriver.h>

MKE_M17_MotorDriver frontWheels; // Module 1 at 0x40
MKE_M17_MotorDriver rearWheels;  // Module 2 at 0x41

void setup() {
  Wire.begin();
  frontWheels.begin(0x40);
  rearWheels.begin(0x41);

  // 4WD Drive Forward
  frontWheels.motorA_CW(200);
  frontWheels.motorB_CW(200);
  rearWheels.motorA_CW(200);
  rearWheels.motorB_CW(200);
}

void loop() {}
```

Example 5: 4-Wheel Mecanum Omnidirectional Robot (4WD / 2 Drivers)

```
#include <Wire.h>
#include <MKE_I2C_MotorDriver.h>

MKE_I2C_MotorDriver driverFront; // Address 0x40 (FL: Motor A, FR: Motor B)
MKE_I2C_MotorDriver driverRear;  // Address 0x41 (RL: Motor A, RR: Motor B)

// Helper function to set individual 4-wheel speeds (-255 to 255)
void set4Wheels(int fl, int fr, int rl, int rr) {
    (fl >= 0) ? driverFront.motorA_CW(fl) : driverFront.motorA_CCW(-fl);
    (fr >= 0) ? driverFront.motorB_CW(fr) : driverFront.motorB_CCW(-fr);
    (rl >= 0) ? driverRear.motorA_CW(rl)  : driverRear.motorA_CCW(-rl);
    (rr >= 0) ? driverRear.motorB_CW(rr)  : driverRear.motorB_CCW(-rr);
}

void setup() {
    Wire.begin();
    driverFront.begin(0x40);
    driverRear.begin(0x41);
}

void loop() {
    // 1. Move Forward (2s)
    set4Wheels(200, 200, 200, 200); delay(2000);
    // 2. Strafe Left (2s)
    set4Wheels(-200, 200, 200, -200); delay(2000);
    // 3. Strafe Right (2s)
    set4Wheels(200, -200, -200, 200); delay(2000);
    // 4. Spin Left / CCW (1.5s)
    set4Wheels(-180, 180, -180, 180); delay(1500);
    // 5. Stop (2s)
    set4Wheels(0, 0, 0, 0); delay(2000);
}
```

7. Programming with BBC micro:bit (MakeCode Extension)

The **MakerEdu I2C Dual DC Motor Driver** provides native support for **BBC micro:bit** with visual block-based programming on Microsoft MakeCode.

1. Installing the MakeCode Extension

1. Open your web browser and navigate to: <https://makecode.microbit.org>
2. Create a **New Project**.
3. Go to **Settings (Gear icon) > Extensions**.
4. Paste the GitHub repository URL:
https://github.com/Khuuxuanngoc/makeCode_MKE_I2C_Motor_Driver_extension_test and hit Enter to add.

2. Interactive Online Tutorials (Step-by-Step Guided Lessons)

Students and educators can click directly on the links below to launch interactive step-by-step tutorial sessions inside Microsoft MakeCode (which automatically loads the MakerEdu Extension and hint system):

-  **Lesson 1: Basic DC Motor Control & Timed Run:**
 [Launch Tutorial 1 in MakeCode](#)
-  **Lesson 2: Battery Voltage Monitoring & Low Power Warning:**
 [Launch Tutorial 2 in MakeCode](#)
-  **Lesson 3: 2-Wheel Differential Robot Programming (2WD Robot):**
 [Launch Tutorial 3 in MakeCode](#)
-  **Lesson 4: Omnidirectional Mecanum Robot Programming (4WD / 2 Drivers):**
 [Launch Tutorial 4 in MakeCode](#)

8. Raw I2C Communication Protocol

For developers implementing drivers on non-Arduino platforms (ESP-IDF, Raspberry Pi Python, STM32 HAL):

1. Write Packet Format (Master → Slave)

- **Total Length:** 6 bytes
- **Structure:** [Region (0x05)] [ModeID (1 byte)] [Payload MSB] [Payload Byte 2] [Payload Byte 1] [Payload LSB]
- **Endianness:** Big-Endian (Most Significant Byte first).
- **Payload Composition:**
 - For standard commands: `payload = speed` (0–255).
 - For timed commands: `payload = ((uint32_t)duration_ms << 16) | (uint32_t)speed`.

2. Read Request Sequence (Master ⇌ Slave)

1. Send 6-byte Write packet with target `ModeID` (payload = 0).
2. Wait at least **200 µs** (`delayMicroseconds(200)`) for the slave MCU to prepare the output buffer.
3. Call I2C Read request for **4 bytes** (`requestFrom(address, 4)`).

4. Combine the 4 received bytes as a 32-bit Big-Endian integer.

3. User Command Map

ModelID (Dec)	Command Name	Type	Payload Format	Description
106	Set_MA_CW	Write	(duration_ms << 16) \ speed	Motor A Clockwise (0–255)
107	Set_MA_CCW	Write	(duration_ms << 16) \ speed	Motor A Counter- Clockwise
110	Set_MA_STOP	Write	0	Immediate Stop Motor A
108	Set_MB_CW	Write	(duration_ms << 16) \ speed	Motor B Clockwise
109	Set_MB_CCW	Write	(duration_ms << 16) \ speed	Motor B Counter- Clockwise
111	Set_MB_STOP	Write	0	Immediate Stop Motor B
112	Get_VIN	Read	Response: uint32_t (mV)	Read Motor Supply Voltage
104	Set_MOTOR_FREQ	Write	frequency_hz	Set PWM frequency (500– 2000Hz)
1	SetAddress	Write	new_address (1–126)	Change I2C Slave Address
10	GetAddress	Read	Response: uint8_t address	Read EEPROM Address
2	Get_ID_Module	Read	Response: uint8_t (3)	Read Device ID
4	Get_FW_Version	Read	Response: uint32_t	Read Firmware Build Date

9. Bluetooth Dabble Gamepad Control with MKE-M15

The module includes a built-in hardware protocol decoder for the **Dabble App** (STEMpedia) over Bluetooth:

1. **Hardware Connection:** Plug the **MKE-M15 Bluetooth UART Module** into the onboard **BLE Port** (VCC, GND, TX, RX).
2. **App Connection:** Open the **Dabble App** on your smartphone and connect via Bluetooth to the MKE-M15 module.
3. **Open Gamepad Module** (supports both Digital Mode and Joystick Mode):
 - **START Button (Activate Control):** After opening the Gamepad screen, **press the START button** so the driver detects active connection ("Connect") and enables motor commands.
 - **SELECT Button (Pause Control):** Press the **SELECT** button to temporarily disable/lock motor controls, keeping the robot safely parked without having to disconnect Bluetooth.
 - **SQUARE Button:** Immediate brake / emergency stop for both motors.
 - **↑ UP Arrow / Joystick Forward:** Both motors drive Forward.
 - **↓ DOWN Arrow / Joystick Backward:** Both motors drive Reverse.
 - **← LEFT Arrow:** Turn Left (differential drive: Motor A backward, Motor B forward).
 - **→ RIGHT Arrow:** Turn Right (differential drive: Motor A forward, Motor B backward).
 - **Analog Joystick:** Smooth proportional speed and steering control mapped to X/Y axes (-7.0 to +7.0).
4. **Signal Loss Failsafe:** If Bluetooth signal is lost, out of range, or the Dabble app is closed, the onboard MCU automatically shuts off PWM signals and halts all motors immediately.

10. Important Notes & Troubleshooting

[!IMPORTANT] **Wire.begin() in Sketch:** The `MKE_I2C_MotorDriver` class does not call `Wire.begin()` internally. Always initialize `Wire.begin()` in your `setup()` function. This allows full customization of custom pins (`Wire.begin(SDA, SCL)`) and higher bus speeds (`Wire.setClock(400000)`).

[!WARNING] Power Configuration & Motor Voltage:

- Always keep **J1 = OFF** when connected to an I2C Master (Arduino/ESP32) and MKE-M15 Bluetooth module.
- **MKE-M17 (L9110):** Rated motor voltage is **6V – 9V DC**.
- **MKE-M18 (BD62130):** Rated motor voltage is **8V – 13V DC**.
- Never short-circuit the motor output terminals.

[!TIP] **Fast Factory Reset:** If you forget the assigned I2C address, press and hold the onboard push button (**S1/SW1**) for **4 seconds**. The module will clear its EEPROM and reset to default address **0x40** (Note: Hardware currently does not have an LED indicator for this event).