Geekobot
Imagine. Build. Innovate.

SPARK KIT

Arduino UNO learning kit
Hands-on project manual

24 PROJECTS · BUILD · LEARN · INNOVATE

A Geekobot exclusive · Beginner to intermediate · No soldering · geekobot.in

What is in the box

- UNO R3 board (CH340) and USB A-B cable
- Sensor Shield v5.0
- 830-point breadboard
- 120 jumper wires (M-M, M-F, F-F)
- 40 x 5 mm LEDs: red, yellow, green, blue
- Resistors: 220 ohm, 330 ohm, 10k (10 each)
- 4 tactile push buttons
- 2 TTP223 touch modules
- Active buzzer and passive buzzer modules
- Sound sensor (digital out)
- LDR light sensor (analog out)
- Rain sensor
- DHT11 temperature and humidity sensor
- IR obstacle sensor
- SG90 micro servo
- 10k potentiometer
- L298N motor driver, DC motor and fan blade
- TM1637 4-digit display
- 2 x 18650 battery holder with switch (cells in the option with cells)

Four levels

Level	Projects	What you do
Level 1 · Getting started	1-5	Upload your first sketch, light LEDs and read buttons and a touch pad.
Level 2 · Light and sound	6-11	Flash patterns, beeps and tunes, a clap sensor and a light sensor.
Level 3 · Sensors and motors	12-21	Rain, temperature, humidity and obstacles; servos, a motor driver and PWM.
Level 4 · Displays and games	22-24	A 4-digit display, a stopwatch and a reaction-time game.

The upload routine (every project)

1. Install the free Arduino IDE 2 from arduino.cc.
2. Connect the UNO with the USB cable.
3. Build the circuit from the project's wiring table, with the USB cable unplugged.
4. Paste the code into a new sketch.
5. Tools > Board > Arduino Uno, then Tools > Port and pick the COM port that appears (install the CH340 driver if none does).
6. Click Upload and wait for "Done uploading".
7. Libraries: Tools > Manage Libraries, when the project lists one.

Safety and good practice

- Always use the stated resistor with every LED.
- Never drive a motor from an Arduino pin: use the L298N and the battery pack (projects 19 and 20).
- Check VCC, GND and signal before powering up; reversed power can damage a module.
- Unplug USB before rewiring.
- 18650 cells: insert them the right way round, never short the holder wires, switch the holder off after use.
- Sensor Shield: each digital pin has its own G (ground), V (5V) and S (signal) pins, so a module plugs into one column with female jumpers.

Every project also has its own page with the same wiring and code, plus troubleshooting: geekobot.in/projects/best/geekobot-spark-kit-projects

CONTENTS

The 24 projects

#	Project	Level	Time
01	Built-in LED blink	Beginner	15 minutes
02	External LED blink	Beginner	20 minutes
03	Traffic light	Beginner	30 minutes
04	Push button LED	Beginner	30 minutes
05	Touch sensor lamp	Beginner	20 minutes
06	Police lights	Beginner	20 minutes
07	Buzzer beep	Beginner	15 minutes
08	Musical buzzer	Beginner	30 minutes
09	Clap detection	Beginner	30 minutes
10	Light detector	Beginner	20 minutes
11	Automatic night lamp	Beginner	30 minutes
12	Rain alarm	Beginner	30 minutes
13	Temperature checker	Beginner	30 minutes
14	Humidity checker	Beginner	20 minutes
15	Servo sweep	Beginner	20 minutes
16	Knob servo	Beginner	30 minutes
17	Obstacle detector	Beginner	20 minutes
18	Intruder alarm	Beginner	20 minutes
19	Fan on/off	Intermediate	45 minutes
20	Smart fan	Intermediate	1 hour
21	LED brightness control	Beginner	20 minutes
22	Digital counter	Intermediate	30 minutes
23	Stopwatch timer	Intermediate	45 minutes
24	Reaction game	Intermediate	45 minutes

PROJECT 01 · LEVEL 1 · GETTING STARTED

Built-in LED blink

Goal: Your very first Arduino program: switch the LED on the UNO board on and off every second.

Every Arduino UNO has a small LED wired to digital pin 13, called LED_BUILTIN in code. This project needs no wiring at all: you upload a five-line sketch that turns that LED on for one second and off for one second, forever. It proves that the IDE, the USB cable and the board all work, which is why every maker starts here.

Online page with troubleshooting: geekobot.in/projects/spark-kit-01-built-in-led-blink

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board

What you learn

- Uploading a sketch to the board
- setup() and loop()
- digitalWrite() outputs
- Timing with delay()

Wiring

Part	Pin	Arduino	Note
Built-in LED	LED_BUILTIN	**D13**	Already on the board: no wiring

Code

```
// Spark Kit 01: built-in LED blink
// Geekobot project page "spark-kit-01-built-in-led-blink"

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);   // pin 13 on the UNO
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // LED on
  delay(1000);                      // wait 1 second
  digitalWrite(LED_BUILTIN, LOW);   // LED off
  delay(1000);
}
```

What you should see: The LED marked L blinks: one second on, one second off.

External LED blink

Goal: Move the blink out to a real LED on the breadboard and meet the resistor that protects it.

An LED on its own would draw far too much current from an Arduino pin and burn out. A 220 ohm resistor in series limits the current to about 15 mA, which is bright and safe. Here pin D12 drives the LED through that resistor, and the same blink code now uses a variable for the pin number, so you can move the LED to another pin by changing one line.

Online page with troubleshooting: geekobot.in/projects/spark-kit-02-external-led-blink

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 1 x Through-Hole LED Pack (Pack of 10)
- 1 x 220 ohm resistor pack
- 1 x Solderless Breadboard
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- Wiring an LED with a current-limiting resistor
- Which LED leg is the anode
- Storing a pin number in a variable

Wiring

Part	Pin	Arduino	Note
LED	Anode (+, long leg)	**D12**	Through the 220 ohm resistor
LED	Cathode (-, short leg)	**GND**	

Code

```
// Spark Kit 02: external LED blink
int led = 12;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

What you should see: The breadboard LED blinks once a second.

Traffic light

Goal: Red, yellow and green LEDs run the same timed sequence as a road signal.

A traffic signal is a fixed sequence of outputs with set times: red for 3 seconds, yellow for 1, green for 3, and round again. Three LEDs on pins D12, D11 and D10, each with its own 220 ohm resistor, model it exactly, and the sketch shows how to switch several outputs together in each step.

Online page with troubleshooting: geekobot.in/projects/spark-kit-03-traffic-light

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 3 x Through-Hole LED Pack (Pack of 10)
- 1 x 220 ohm resistor pack
- 1 x Solderless Breadboard
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- Controlling several outputs
- Building a timed sequence
- Modelling a real system in code

Wiring

Part	Pin	Arduino	Note
Red LED	Anode (+)	**D12**	Through 220 ohm
Yellow LED	Anode (+)	**D11**	Through 220 ohm
Green LED	Anode (+)	**D10**	Through 220 ohm
All three LEDs	Cathode (-)	**GND**	Use the breadboard's blue rail

Code

```
// Spark Kit 03: traffic light
int red = 12;
int yellow = 11;
int green = 10;

void setup() {
  pinMode(red, OUTPUT);
  pinMode(yellow, OUTPUT);
  pinMode(green, OUTPUT);
}

void light(int r, int y, int g, int ms) {
  digitalWrite(red, r);
  digitalWrite(yellow, y);
  digitalWrite(green, g);
  delay(ms);
}

void loop() {
  light(HIGH, LOW, LOW, 3000);  // red 3 s
  light(LOW, HIGH, LOW, 1000);  // yellow 1 s
  light(LOW, LOW, HIGH, 3000);  // green 3 s
}
```

What you should see: Red 3 s, yellow 1 s, green 3 s, repeating.

Push button LED

Goal: Read a push button and light an LED only while it is held down.

A button is the simplest digital input. One side goes to 5V, the other to pin D2, and a 10k resistor from D2 to GND (a pull-down) holds the pin at LOW when nobody presses. Pressing connects 5V and the pin reads HIGH. The sketch reads the pin with digitalRead() and copies the state to an LED on D3.

Online page with troubleshooting: geekobot.in/projects/spark-kit-04-push-button-led

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 1 x Through-Hole LED Pack (Pack of 10)
- 1 x Tactile Push Button Switch
- 2 x 220 ohm and 10k resistor pack
- 1 x Solderless Breadboard
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- Reading a digital input
- Why an input needs a pull-down resistor
- if / else decisions

Wiring

Part	Pin	Arduino	Note
LED	Anode (+)	**D3**	Through 220 ohm
LED	Cathode (-)	**GND**	
Push button	Leg 1	**5V**	
Push button	Leg 2	**D2**	Also 10k resistor from this leg to GND (pull-down)

Code

```
// Spark Kit 04: push button LED
int led = 3;
int button = 2;

void setup() {
  pinMode(led, OUTPUT);
  pinMode(button, INPUT);   // 10k pull-down on the breadboard
}

void loop() {
  int state = digitalRead(button);
  if (state == HIGH) {
    digitalWrite(led, HIGH);
  } else {
    digitalWrite(led, LOW);
  }
}
```

What you should see: Press: LED on. Release: LED off.

Touch sensor lamp

Goal: A capacitive touch pad that lights an LED while your finger rests on it.

The TTP223 module senses the small change in capacitance when a finger comes near its pad and sets its S pin HIGH. To the Arduino that looks exactly like a button, so the code is the same as project 04: read D11, copy the state to the LED on D4. No moving parts, no pull-down needed, and it works through thin plastic.

Online page with troubleshooting: geekobot.in/projects/spark-kit-05-touch-sensor-lamp

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 1 x Capacitive Touch Sensor Module
- 1 x Through-Hole LED Pack (Pack of 10)
- 1 x 220 ohm resistor pack
- 1 x Solderless Breadboard
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- Reading a sensor module
- Capacitive touch basics
- Modules with a built-in digital output

Wiring

Part	Pin	Arduino	Note
Touch sensor	VCC / GND	**5V / GND**	
Touch sensor	S (signal)	**D11**	
LED	Anode (+)	**D4**	Through 220 ohm
LED	Cathode (-)	**GND**	

Code

```
// Spark Kit 05: touch sensor lamp
int touch = 11;
int led = 4;

void setup() {
  pinMode(touch, INPUT);
  pinMode(led, OUTPUT);
}

void loop() {
  int state = digitalRead(touch);
  digitalWrite(led, state == HIGH ? HIGH : LOW);
}
```

What you should see: Touch the pad: LED on. Lift your finger: LED off.

Police lights

Goal: Three quick red flashes, three quick blue flashes: the pattern of an emergency vehicle.

Fast, repeated flashes are just a short on-off pattern run several times. A for loop repeats the 150 ms flash three times for the red LED on D8, then three times for the blue LED on D9, and loop() starts over. Change the count or the timing to make your own pattern.

Online page with troubleshooting: geekobot.in/projects/spark-kit-06-police-light-effect

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 2 x Through-Hole LED Pack (Pack of 10)
- 1 x 330 ohm resistor pack
- 1 x Solderless Breadboard
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- Digital outputs
- for loops
- Fast timing for visual effects

Wiring

Part	Pin	Arduino	Note
Red LED	Anode (+)	**D8**	Through 330 ohm
Blue LED	Anode (+)	**D9**	Through 330 ohm
Both LEDs	Cathode (-)	**GND**	

Code

```
// Spark Kit 06: police light effect
int redLED = 8;
int blueLED = 9;

void flash(int pin, int times) {
  for (int i = 0; i < times; i++) {
    digitalWrite(pin, HIGH);
    delay(150);
    digitalWrite(pin, LOW);
    delay(150);
  }
}

void setup() {
  pinMode(redLED, OUTPUT);
  pinMode(blueLED, OUTPUT);
}

void loop() {
  flash(redLED, 3);
  flash(blueLED, 3);
}
```

What you should see: Three red flashes, three blue flashes, repeating.

Buzzer beep

Goal: Switch an active buzzer on and off to make a steady beep, then a fast alarm.

An active buzzer has its own oscillator inside: give its signal pin HIGH and it sounds a fixed tone, give it LOW and it stops. That makes it the easiest way to add sound to a project. The sketch beeps half a second on, half a second off; change the delay to 100 ms and you have an alarm.

Online page with troubleshooting: geekobot.in/projects/spark-kit-07-buzzer-beep

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 1 x PCB Mounted Active Buzzer Module
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- Driving an active buzzer
- Sound generation from a digital pin
- Changing a rhythm with timing

Wiring

Part	Pin	Arduino	Note
Active buzzer	VCC	**5V**	
Active buzzer	GND	**GND**	
Active buzzer	I/O (signal)	**D11**	

Code

```
// Spark Kit 07: buzzer beep
int buzzer = 11;
int beepMs = 500;   // try 100 for a fast alarm

void setup() {
  pinMode(buzzer, OUTPUT);
}

void loop() {
  digitalWrite(buzzer, HIGH);
  delay(beepMs);
  digitalWrite(buzzer, LOW);
  delay(beepMs);
}
```

What you should see: A steady half-second beep.

Musical buzzer

Goal: Play a short tune by sending note frequencies to a passive buzzer with tone().

Every musical note is a frequency: middle C is 262 Hz, A is 440 Hz. The Arduino tone() function makes a square wave at any frequency on a pin, and a passive buzzer turns it into sound. A list of notes and a list of lengths are played one by one, then the tune pauses for 3 seconds and repeats. This needs the passive buzzer from the kit: an active buzzer only knows one tone.

Online page with troubleshooting: geekobot.in/projects/spark-kit-08-musical-buzzer

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 1 x Buzzer Module 3.3–5V
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- tone() and noTone()
- Musical note frequencies
- Arrays of notes and durations

Wiring

Part	Pin	Arduino	Note
Passive buzzer	VCC	**5V**	
Passive buzzer	GND	**GND**	
Passive buzzer	I/O (signal)	**D11**	

Code

```
// Spark Kit 08: musical buzzer (passive buzzer on D11)
int buzzer = 11;

// "Twinkle twinkle" opening: C C G G A A G
int melody[]   = {262, 262, 392, 392, 440, 440, 392};
int duration[] = {400, 400, 400, 400, 400, 400, 800};

void setup() {}

void loop() {
  for (int i = 0; i < 7; i++) {
    tone(buzzer, melody[i], duration[i]);
    delay(duration[i] * 1.3);   // short gap between notes
  }
  noTone(buzzer);
  delay(3000);
}
```

What you should see: The tune plays, pauses for 3 seconds and repeats.

Clap detection

Goal: A sound sensor spots a clap and lights an LED the instant it hears one.

The sound module has a microphone, an LM393 comparator and a small trimmer. When the sound is louder than the level you set, its DO pin goes LOW. The Arduino watches DO on D7 and lights the LED on D2 for a moment each time it sees a clap. Setting the trimmer is the real lesson: too sensitive and talking triggers it, too dull and claps are missed.

Online page with troubleshooting: geekobot.in/projects/spark-kit-09-clap-detection

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 1 x Generic LM393 Sound Detection Sensor Module – Digital Output
- 1 x Through-Hole LED Pack (Pack of 10)
- 1 x 220 ohm resistor pack
- 1 x Solderless Breadboard
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- Digital sensors with an adjustable threshold
- Event detection
- Smart-home basics

Wiring

Part	Pin	Arduino	Note
Sound sensor	VCC / GND	**5V / GND**	
Sound sensor	DO	**D7**	Goes LOW on a loud sound
LED	Anode (+)	**D2**	Through 220 ohm
LED	Cathode (-)	**GND**	

Code

```
// Spark Kit 09: clap detection
int sound = 7;   // DO of the sound sensor
int led = 2;

void setup() {
  pinMode(sound, INPUT);
  pinMode(led, OUTPUT);
}

void loop() {
  if (digitalRead(sound) == LOW) {   // most modules pull DO LOW on a loud sound
    digitalWrite(led, HIGH);
    delay(500);                      // keep it lit so you can see it
  } else {
    digitalWrite(led, LOW);
  }
}
```

What you should see: Clap: the LED flashes for half a second.

Light detector

Goal: Read the light level with an LDR module and watch the numbers change in the Serial Monitor.

An LDR's resistance falls as light rises. The module turns that into a voltage between 0 and 5 V, and analogRead() on A0 turns the voltage into a number from 0 to 1023. Printing it every half second shows how the Arduino's ADC sees the world: cover the sensor, shine a phone torch on it, and note the two numbers. Project 11 uses them as its threshold.

Online page with troubleshooting: geekobot.in/projects/spark-kit-10-light-detector

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 1 x Generic Photosensitive Resistor Sensor Module for Arduino – Light Dependent Resistor (LDR)
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- Analog inputs
- Analog-to-digital conversion (ADC)
- Monitoring sensor data on the Serial Monitor

Wiring

Part	Pin	Arduino	Note
LDR module	VCC / GND	**5V / GND**	
LDR module	AO / S (analog out)	**A0**	

Code

```
// Spark Kit 10: light detector
int ldr = A0;

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

void loop() {
  int value = analogRead(ldr);   // 0..1023
  Serial.print("Light level: ");
  Serial.println(value);
  delay(500);
}
```

What you should see: Open the Serial Monitor at 9600 baud: the number changes when you cover the sensor or shine a torch on it.

Automatic night lamp

Goal: An LED that switches itself on when the room gets dark and off when it is bright.

This turns project 10's numbers into a decision. The sketch reads the LDR on A0, compares it with a threshold and switches the LED on D2. The threshold and the direction live in two constants at the top, so you set them once from your Serial Monitor readings and the lamp works with any LDR module.

Online page with troubleshooting: geekobot.in/projects/spark-kit-11-automatic-night-lamp

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 1 x Generic Photosensitive Resistor Sensor Module for Arduino – Light Dependent Resistor (LDR)
- 1 x Through-Hole LED Pack (Pack of 10)
- 1 x 220 ohm resistor pack
- 1 x Solderless Breadboard
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- Analog thresholds
- Automatic control
- Calibrating a sensor from real readings

Wiring

Part	Pin	Arduino	Note
LDR module	VCC / GND	**5V / GND**	
LDR module	AO / S (analog out)	**A0**	
LED	Anode (+)	**D2**	Through 220 ohm
LED	Cathode (-)	**GND**	

Code

```
// Spark Kit 11: automatic night lamp
int ldr = A0;
int led = 2;
int THRESHOLD = 300;          // halfway between your bright and dark readings
bool DARK_IS_LOW = true;      // set false if your module reads higher in the dark

void setup() {
  pinMode(led, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  int value = analogRead(ldr);
  Serial.println(value);
  bool dark = DARK_IS_LOW ? (value < THRESHOLD) : (value > THRESHOLD);
  digitalWrite(led, dark ? HIGH : LOW);
  delay(200);
}
```

What you should see: Cover the sensor: LED on. Uncover: LED off.

Rain alarm

Goal: Sound a buzzer as soon as water drops land on the rain plate.

The rain plate has two comb-shaped tracks. Water bridges them and lowers the resistance, and the LM393 board turns that into a digital signal: DO goes LOW when wet. The Arduino reads DO on D2 and sounds the active buzzer on D8 while the plate is wet. Use it to warn you to bring the washing in.

Online page with troubleshooting: geekobot.in/projects/spark-kit-12-rain-alarm

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 1 x Generic Raindrops Detection Sensor Module – LM393 Comparator, Analog/Digital Output
- 1 x PCB Mounted Active Buzzer Module
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- Water detection
- Active-low digital sensors
- A simple alarm system

Wiring

Part	Pin	Arduino	Note
Rain sensor board	VCC / GND	**5V / GND**	
Rain sensor board	DO	**D2**	LOW when wet
Active buzzer	VCC / GND	**5V / GND**	
Active buzzer	I/O (signal)	**D8**	

Code

```
// Spark Kit 12: rain alarm
int rain = 2;     // DO, LOW when wet
int buzzer = 8;

void setup() {
  pinMode(rain, INPUT);
  pinMode(buzzer, OUTPUT);
}

void loop() {
  if (digitalRead(rain) == LOW) {
    digitalWrite(buzzer, HIGH);
  } else {
    digitalWrite(buzzer, LOW);
  }
  delay(100);
}
```

What you should see: Drip a little water on the plate: the buzzer sounds until you wipe it dry.

Temperature checker

Goal: Read the room temperature from a DHT11 sensor and print it every 2 seconds.

The DHT11 measures temperature and humidity and sends both over a single data wire. Decoding that signal by hand is fiddly, so we use a library: two lines set it up and dht.readTemperature() returns degrees Celsius as a decimal number. This is your first project that uses a library, and the first that works with float values.

Online page with troubleshooting: geekobot.in/projects/spark-kit-13-temperature-checker

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 1 x Generic DHT11 Digital Temperature & Humidity Sensor Module
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- Installing and using a library
- Reading a digital sensor
- Working with float values

Wiring

Part	Pin	Arduino	Note
DHT11	VCC (+)	**5V**	
DHT11	GND (-)	**GND**	
DHT11	DATA (S)	**D2**	

Code

```
// Spark Kit 13: temperature checker
#include <DHT.h>
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
}

void loop() {
  float t = dht.readTemperature();   // degrees C
  if (isnan(t)) {
    Serial.println("DHT11 not found: check the wiring");
  } else {
    Serial.print("Temperature: ");
    Serial.print(t, 1);
    Serial.println(" C");
  }
  delay(2000);   // the DHT11 needs about 2 s between reads
}
```

What you should see: The Serial Monitor (9600 baud) prints the temperature every 2 seconds; hold the sensor and it rises.

Humidity checker

Goal: Read the relative humidity of the room from the same DHT11 sensor.

The DHT11 also measures relative humidity: how much water vapour the air holds compared with the most it could hold at that temperature. Same wiring and library as project 13; only the read call changes, to dht.readHumidity(). Breathe on the sensor and watch the number climb.

Online page with troubleshooting: geekobot.in/projects/spark-kit-14-humidity-checker

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 1 x Generic DHT11 Digital Temperature & Humidity Sensor Module
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- Reading a second value from one sensor
- What relative humidity means
- Environmental monitoring

Wiring

Part	Pin	Arduino	Note
DHT11	VCC (+)	**5V**	
DHT11	GND (-)	**GND**	
DHT11	DATA (S)	**D2**	

Code

```
// Spark Kit 14: humidity checker
#include <DHT.h>
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
}

void loop() {
  float h = dht.readHumidity();   // percent
  if (isnan(h)) {
    Serial.println("DHT11 not found: check the wiring");
  } else {
    Serial.print("Humidity: ");
    Serial.print(h, 0);
    Serial.println(" %");
  }
  delay(2000);
}
```

What you should see: The Serial Monitor prints the humidity every 2 seconds; breathe on the sensor and it rises.

Servo sweep

Goal: Swing an SG90 servo arm smoothly from 0 to 180 degrees and back, over and over.

A hobby servo turns to the angle you ask for and holds it. It reads the angle from the width of a pulse sent 50 times a second, and the built-in Servo library makes those pulses for you: myServo.write(90) means 90 degrees. Two for loops step the angle one degree at a time, which gives a smooth sweep.

Online page with troubleshooting: geekobot.in/projects/spark-kit-15-servo-motor-sweep

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 1 x TowerPro SG90 9g Micro Servo (180°)
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- PWM pulses and servo control
- Servo positioning with the Servo library
- Robotics basics

Wiring

Part	Pin	Arduino	Note
Servo	Brown / black	**GND**	
Servo	Red	**5V**	
Servo	Orange / yellow (signal)	**D9**	

Code

```
// Spark Kit 15: servo sweep
#include <Servo.h>
Servo myServo;

void setup() {
  myServo.attach(9);
}

void loop() {
  for (int pos = 0; pos <= 180; pos++) {
    myServo.write(pos);
    delay(15);
  }
  for (int pos = 180; pos >= 0; pos--) {
    myServo.write(pos);
    delay(15);
  }
}
```

What you should see: The servo arm sweeps from one end to the other and back, about 3 seconds each way.

Knob servo

Goal: Turn a potentiometer and the servo arm follows your hand from 0 to 180 degrees.

A potentiometer is a voltage divider you turn by hand. Its middle pin gives 0 to 5 V, read on A0 as 0 to 1023. map() rescales that to 0 to 180 degrees, and the servo goes there. This read-scale-act pattern is how joysticks, steering wheels and robot arms are controlled.

Online page with troubleshooting: geekobot.in/projects/spark-kit-16-knob-controlled-servo

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 1 x 16mm Rotary Potentiometer (Linear)
- 1 x TowerPro SG90 9g Micro Servo (180°)
- 1 x Solderless Breadboard
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- Analog input from a potentiometer
- Rescaling with map()
- Human-machine control

Wiring

Part	Pin	Arduino	Note
Potentiometer	Outer leg 1	**5V**	
Potentiometer	Outer leg 2	**GND**	
Potentiometer	Middle leg (wiper)	**A0**	
Servo	Brown / black	**GND**	
Servo	Red	**5V**	
Servo	Orange / yellow (signal)	**D9**	

Code

```
// Spark Kit 16: knob-controlled servo
#include <Servo.h>
Servo myServo;
int pot = A0;

void setup() {
  myServo.attach(9);
}

void loop() {
  int value = analogRead(pot);              // 0..1023
  int angle = map(value, 0, 1023, 0, 180);  // 0..180 degrees
  myServo.write(angle);
  delay(15);
}
```

What you should see: Turn the knob slowly: the servo arm follows it across its full range.

Obstacle detector

Goal: An LED lights whenever something comes close to the infrared sensor.

The IR module shines invisible infrared light forward and watches for its reflection. When an object comes within range (about 2 to 30 cm, set by the trimmer) its DO pin goes LOW. The Arduino reads that on D2 and lights the LED on D8. Robot cars use the same sensor to stop before a wall.

Online page with troubleshooting: geekobot.in/projects/spark-kit-17-obstacle-detector

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 1 x Generic Infrared Obstacle Avoidance Sensor Module (Active Low) – LM393 Comparator, 2-30cm Range
- 1 x Through-Hole LED Pack (Pack of 10)
- 1 x 330 ohm resistor pack
- 1 x Solderless Breadboard
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- Object sensing
- Reflective IR sensors
- Active-low outputs

Wiring

Part	Pin	Arduino	Note
IR sensor	VCC / GND	**5V / GND**	
IR sensor	OUT / DO	**D2**	LOW when an object is near
LED	Anode (+)	**D8**	Through 330 ohm
LED	Cathode (-)	**GND**	

Code

```
// Spark Kit 17: obstacle detector
int ir = 2;    // LOW when an object is near
int led = 8;

void setup() {
  pinMode(ir, INPUT);
  pinMode(led, OUTPUT);
}

void loop() {
  digitalWrite(led, digitalRead(ir) == LOW ? HIGH : LOW);
}
```

What you should see: Bring your hand towards the sensor: the LED lights.

Intruder alarm

Goal: The buzzer sounds the moment someone passes in front of the IR sensor.

Swap project 17's LED for the active buzzer and you have a doorway alarm. Point the IR sensor across a door or a drawer opening; when someone breaks the beam, DO on D2 goes LOW and the buzzer on D8 sounds. A short hold time keeps the alarm going for a second after each detection so it is easy to hear.

Online page with troubleshooting: geekobot.in/projects/spark-kit-18-intruder-alarm

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 1 x Generic Infrared Obstacle Avoidance Sensor Module (Active Low) – LM393 Comparator, 2-30cm Range
- 1 x PCB Mounted Active Buzzer Module
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- Security system logic
- Detection with a hold time
- Reusing a sensor for a new job

Wiring

Part	Pin	Arduino	Note
IR sensor	VCC / GND	**5V / GND**	
IR sensor	OUT / DO	**D2**	
Active buzzer	VCC / GND	**5V / GND**	
Active buzzer	I/O (signal)	**D8**	

Code

```
// Spark Kit 18: intruder alarm
int ir = 2;
int buzzer = 8;

void setup() {
  pinMode(ir, INPUT);
  pinMode(buzzer, OUTPUT);
}

void loop() {
  if (digitalRead(ir) == LOW) {   // something detected
    digitalWrite(buzzer, HIGH);
    delay(1000);                  // hold the alarm for 1 second
  } else {
    digitalWrite(buzzer, LOW);
  }
}
```

What you should see: Wave a hand in front of the sensor: the buzzer sounds for about a second.

Fan on/off

Goal: Press a button to run a small DC fan, driven safely through an L298N motor driver.

A motor needs far more current than an Arduino pin can give, so it gets its own battery and an L298N driver. The Arduino only sends logic signals: IN1 HIGH and IN2 LOW spin the motor, both LOW stop it. The button on D11 uses the UNO's internal pull-up, so it reads LOW when pressed and needs no resistor.

Online page with troubleshooting: geekobot.in/projects/spark-kit-19-fan-on-off

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 1 x L298N Dual H-Bridge Motor Driver Module
- 1 x Hobby DC Motor (130 / 180 Size)
- 1 x DC Motor Fan Blade (Toy Propeller)
- 1 x Tactile Push Button Switch
- 1 x 18650 Battery Holder
- 2 x 18650 Li-ion Rechargeable Cell
- 1 x Solderless Breadboard
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- Switching a motor through a driver
- Separate power for motors with a common ground
- INPUT_PULLUP buttons

Wiring

Part	Pin	Arduino	Note
DC motor	Two wires	**OUT1 and OUT2 of the L298N**	
Battery pack	+ (red)	**L298N 12V terminal**	
Battery pack	- (black)	**L298N GND terminal**	
L298N	GND terminal	**Arduino GND**	Common ground: shares the terminal with the battery -
L298N	IN1	**D4**	
L298N	IN2	**D3**	
L298N	ENA jumper	**Leave fitted**	Full speed; the 5V-EN jumper also stays on
Push button	One leg	**D11**	INPUT_PULLUP: no resistor needed
Push button	Other leg	**GND**	

> The motor runs from the 2 x 18650 pack (7.4 V) through the L298N; the Arduino runs from USB. The two grounds must be joined.

> Li-ion cells: insert them the right way round, never short the holder's wires, and switch the holder off when you finish.

> Keep fingers and hair away from the spinning fan blade.

Code

```
// Spark Kit 19: fan on/off with L298N
int IN1 = 4;
int IN2 = 3;
int button = 11;   // to GND, uses the internal pull-up

void setup() {
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(button, INPUT_PULLUP);
}

void loop() {
  if (digitalRead(button) == LOW) {   // pressed
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
  } else {
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
  }
}
```

What you should see: Hold the button: the fan spins. Let go: it stops.

Smart fan

Goal: The fan switches itself on when the room goes above 30 degrees C and off when it cools.

This joins a sensor to an actuator: the DHT11 reads the temperature every 2 seconds and the Arduino turns the L298N fan on above 30 degrees C and off when it falls back. It is a closed loop, the same idea as a thermostat. A 1-degree gap between on and off (hysteresis) stops the fan chattering right at the limit.

Online page with troubleshooting: geekobot.in/projects/spark-kit-20-smart-fan

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 1 x Generic DHT11 Digital Temperature & Humidity Sensor Module
- 1 x L298N Dual H-Bridge Motor Driver Module
- 1 x Hobby DC Motor (130 / 180 Size)
- 1 x DC Motor Fan Blade (Toy Propeller)
- 1 x 18650 Battery Holder
- 2 x 18650 Li-ion Rechargeable Cell
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- Automation from a sensor reading
- Combining a sensor with an actuator
- Closed-loop control and hysteresis

Wiring

Part	Pin	Arduino	Note
DHT11	VCC / GND	**5V / GND**	
DHT11	DATA	**D2**	
L298N	IN1	**D8**	
L298N	IN2	**D9**	
DC motor	Two wires	**OUT1 and OUT2 of the L298N**	
Battery pack	+ / -	**L298N 12V / GND terminals**	
L298N	GND terminal	**Arduino GND**	Common ground
L298N	ENA jumper	**Leave fitted**	

The motor runs from the 2 x 18650 pack through the L298N; the Arduino and DHT11 run from USB. Join the grounds.

Li-ion cells: insert them the right way round, never short the holder's wires, and switch the holder off when you finish.

Keep fingers and hair away from the spinning fan blade.

Code

```
// Spark Kit 20: smart fan
#include <DHT.h>
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

int IN1 = 8;
int IN2 = 9;
float ON_AT = 30.0;    // fan on above this (degrees C)
float OFF_AT = 29.0;   // fan off below this
bool fanOn = false;

void setup() {
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  Serial.begin(9600);
  dht.begin();
}

void loop() {
  float t = dht.readTemperature();
  if (!isnan(t)) {
    if (t > ON_AT) fanOn = true;
    if (t < OFF_AT) fanOn = false;
    Serial.print("Temp: "); Serial.print(t, 1);
    Serial.println(fanOn ? " C  fan ON" : " C  fan OFF");
  }
  digitalWrite(IN1, fanOn ? HIGH : LOW);
  digitalWrite(IN2, LOW);
  delay(2000);
}
```

What you should see: Warm the DHT11 in your hand above the limit: the fan starts; let it cool and it stops.

LED brightness control

Goal: Turn a knob to dim and brighten an LED smoothly using PWM.

An Arduino pin is either on or off, but switching it very fast and changing how long it stays on each time (PWM) makes an LED look dimmer or brighter. analogWrite() does that on the PWM pins (3, 5, 6, 9, 10, 11) with a value from 0 to 255. The potentiometer on A0 sets the value, so the knob becomes a dimmer.

Online page with troubleshooting: geekobot.in/projects/spark-kit-21-led-brightness-control

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 1 x 16mm Rotary Potentiometer (Linear)
- 1 x Through-Hole LED Pack (Pack of 10)
- 1 x 220 ohm resistor pack
- 1 x Solderless Breadboard
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- PWM with analogWrite()
- Mapping 0-1023 to 0-255
- Smooth output control

Wiring

Part	Pin	Arduino	Note
Potentiometer	Outer legs	**5V and GND**	
Potentiometer	Middle leg (wiper)	**A0**	
LED	Anode (+)	**D9 (PWM)**	Through 220 ohm
LED	Cathode (-)	**GND**	

Code

```
// Spark Kit 21: LED brightness control
int pot = A0;
int led = 9;   // PWM pin

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  int value = analogRead(pot);                    // 0..1023
  int brightness = map(value, 0, 1023, 0, 255);  // 0..255
  analogWrite(led, brightness);
  delay(10);
}
```

What you should see: Turn the knob: the LED fades smoothly from off to full brightness.

Digital counter

Goal: Count from 0000 to 9999 on a 4-digit 7-segment display, then start again.

The TM1637 module drives four 7-segment digits over just two wires, CLK and DIO. With the TM1637Display library, display.showNumberDec(n, true) shows any number up to 9999 with leading zeros. A counter that adds one every 100 ms is the simplest way to see it working.

Online page with troubleshooting: geekobot.in/projects/spark-kit-22-digital-counter

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 1 x TM1637 4-Digit 7-Segment Display Module
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- Driving a display with a library
- Counting logic and wrap-around
- Two-wire display interfaces

Wiring

Part	Pin	Arduino	Note
Display	VCC	**5V**	
Display	GND	**GND**	
Display	CLK	**D2**	
Display	DIO	**D3**	

Code

```
// Spark Kit 22: digital counter
#include <TM1637Display.h>
#define CLK 2
#define DIO 3
TM1637Display display(CLK, DIO);

int count = 0;

void setup() {
  display.setBrightness(5);   // 0..7
}

void loop() {
  display.showNumberDec(count, true);   // true = leading zeros
  count++;
  if (count > 9999) count = 0;
  delay(100);
}
```

What you should see: The display counts 0000, 0001, 0002 and keeps going.

Stopwatch timer

Goal: A stopwatch on the 4-digit display, counting in MM:SS with the colon blinking.

Counting seconds with delay(1000) drifts, because the code itself takes time. This stopwatch uses millis(), the board's own millisecond clock, and works out minutes and seconds with division and modulo: 75 seconds is 75 / 60 = 1 minute and 75 % 60 = 15 seconds, shown as 01:15. The colon blinks every half second.

Online page with troubleshooting: geekobot.in/projects/spark-kit-23-stopwatch-timer

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 1 x TM1637 4-Digit 7-Segment Display Module
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- Timing with millis()
- Division and modulo
- Formatting MM:SS on a display

Wiring

Part	Pin	Arduino	Note
Display	VCC	**5V**	
Display	GND	**GND**	
Display	CLK	**D2**	
Display	DIO	**D3**	

Code

```
// Spark Kit 23: stopwatch MM:SS
#include <TM1637Display.h>
#define CLK 2
#define DIO 3
TM1637Display display(CLK, DIO);

unsigned long start;

void setup() {
  display.setBrightness(5);
  start = millis();
}

void loop() {
  unsigned long ms = millis() - start;
  unsigned long sec = ms / 1000;
  int mm = (sec / 60) % 100;
  int ss = sec % 60;
  bool colon = (ms % 1000) < 500;                 // blink the colon
  display.showNumberDecEx(mm * 100 + ss, colon ? 0b01000000 : 0, true);
  delay(50);
}
```

What you should see: The display shows 00:00 and counts up in minutes and seconds.

Reaction game

Goal: Wait for the LED, hit the button as fast as you can, and see your reaction time in milliseconds.

The Arduino waits a random 2 to 5 seconds so you cannot guess, lights the LED and starts a millis() timer. When you press the button the timer stops, the buzzer beeps and your time prints in the Serial Monitor. Press too early and it calls a false start. Under 200 ms is superhuman; 200 to 300 ms is excellent.

Online page with troubleshooting: geekobot.in/projects/spark-kit-24-reaction-time-game

Parts

- 1 x Uno R3 CH340G ATmega328p Compatible Development Board
- 1 x Through-Hole LED Pack (Pack of 10)
- 1 x 330 ohm resistor pack
- 1 x Tactile Push Button Switch
- 1 x PCB Mounted Active Buzzer Module
- 1 x Solderless Breadboard
- 1 x Dupont Jumper Wires (40 pcs)
- 1 x Arduino Sensor Shield V5 Expansion Board

What you learn

- millis() timing
- Random numbers with random()
- Reading inputs in real time
- Putting a whole game together

Wiring

Part	Pin	Arduino	Note
LED	Anode (+)	**D2**	Through 330 ohm
LED	Cathode (-)	**GND**	
Push button	One leg	**D3**	INPUT_PULLUP
Push button	Other leg	**GND**	
Active buzzer	VCC / GND	**5V / GND**	
Active buzzer	I/O (signal)	**D8**	

Code

```
// Spark Kit 24: reaction time game
int led = 2;
int button = 3;   // to GND, internal pull-up
int buzzer = 8;

void setup() {
  pinMode(led, OUTPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(button, INPUT_PULLUP);
  Serial.begin(9600);
  randomSeed(analogRead(A0));   // unconnected pin gives a random seed
  Serial.println("Get ready...");
}

void loop() {
  digitalWrite(led, LOW);
  unsigned long wait = random(2000, 5000);
  unsigned long t0 = millis();
  while (millis() - t0 < wait) {
    if (digitalRead(button) == LOW) {   // pressed too early
      Serial.println("False start! Wait for the LED.");
      delay(1000);
      return;
    }
  }
  digitalWrite(led, HIGH);
  unsigned long startTime = millis();
  while (digitalRead(button) == HIGH) {}   // wait for the press
  unsigned long reactionTime = millis() - startTime;

  digitalWrite(buzzer, HIGH);
  delay(150);
  digitalWrite(buzzer, LOW);
  Serial.print("Reaction Time: ");
  Serial.print(reactionTime);
  Serial.println(" ms");
  delay(2000);
}
```

What you should see: After a random wait the LED lights; press the button and your time in ms prints in the Serial Monitor.

WHAT NEXT

Keep building

Every project in this book follows one pattern: read an input, make a decision, drive an output. A night lamp, an intruder alarm and a smart fan are the same idea with different parts. Now remix them:

- Add the buzzer from the intruder alarm to the rain sensor, or show the light level on the 4-digit display.
- Add a start and stop button to the stopwatch.
- Make the smart fan speed up as it gets hotter with PWM on the L298N ENA pin.
- Try new modules: an ultrasonic distance sensor, an OLED screen, Bluetooth.

More than 1,700 free project guides with parts lists, wiring and code: **geekobot.in/projects**

Geekobot
Imagine. Build. Innovate.

Build. Learn. Innovate.