**How to use**

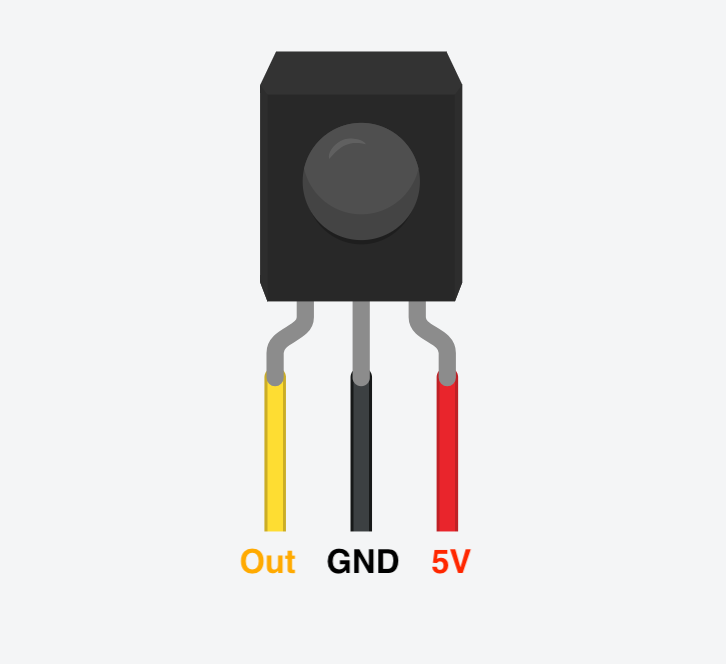
**TL 1838 Infrared Receiver**

**1. Install Library**

Download Ken Shirriff’s IR remote library found on github: <https://github.com/z3t0/Arduino-IRremote/releases/download/MAJOR/IRremote.zip>

Add libraries through Sketch -> Include Library -> Add .ZIP Library

**2. Wiring**



Out → Any digital pin. Example code uses pin 11.

**3. Decode signal from IR remote**

If this code doesn’t work, you can also download [**HERE**](https://drive.google.com/drive/folders/1nlDM8dbkJDqSv1CxR50lg8yj5gm7Dnr-?usp=sharing).

|  |
| --- |
| //------------------------------------------------------------------------------ // Include the IRremote library header // #include <IRremote.h>  //------------------------------------------------------------------------------ // Tell IRremote which Arduino pin is connected to the IR Receiver (TSOP4838) // int recvPin = 11; IRrecv irrecv(recvPin);  //+============================================================================= // Configure the Arduino // void setup ( ) {  Serial.begin(9600); // Status message will be sent to PC at 9600 baud  irrecv.enableIRIn(); // Start the receiver }  //+============================================================================= // Display IR code // void ircode (decode\_results \*results) {  // Panasonic has an Address  if (results->decode\_type == PANASONIC) {  Serial.print(results->address, HEX);  Serial.print(":");  }   // Print Code  Serial.print(results->value, HEX); }  //+============================================================================= // Display encoding type // void encoding (decode\_results \*results) {  switch (results->decode\_type) {  default:  case UNKNOWN: Serial.print("UNKNOWN"); break ;  case NEC: Serial.print("NEC"); break ;  case SONY: Serial.print("SONY"); break ;  case RC5: Serial.print("RC5"); break ;  case RC6: Serial.print("RC6"); break ;  case DISH: Serial.print("DISH"); break ;  case SHARP: Serial.print("SHARP"); break ;  case JVC: Serial.print("JVC"); break ;  case SANYO: Serial.print("SANYO"); break ;  case MITSUBISHI: Serial.print("MITSUBISHI"); break ;  case SAMSUNG: Serial.print("SAMSUNG"); break ;  case LG: Serial.print("LG"); break ;  case WHYNTER: Serial.print("WHYNTER"); break ;  case AIWA\_RC\_T501: Serial.print("AIWA\_RC\_T501"); break ;  case PANASONIC: Serial.print("PANASONIC"); break ;  case DENON: Serial.print("Denon"); break ;  } }  //+============================================================================= // Dump out the decode\_results structure. // void dumpInfo (decode\_results \*results) {  // Check if the buffer overflowed  if (results->overflow) {  Serial.println("IR code too long. Edit IRremoteInt.h and increase RAWBUF");  return;  }   // Show Encoding standard  Serial.print("Encoding : ");  encoding(results);  Serial.println("");   // Show Code & length  Serial.print("Code : ");  ircode(results);  Serial.print(" (");  Serial.print(results->bits, DEC);  Serial.println(" bits)"); }  //+============================================================================= // Dump out the decode\_results structure. // void dumpRaw (decode\_results \*results) {  // Print Raw data  Serial.print("Timing[");  Serial.print(results->rawlen - 1, DEC);  Serial.println("]: ");   for (int i = 1; i < results->rawlen; i++) {  unsigned long x = results->rawbuf[i] \* USECPERTICK;  if (!(i & 1)) { // even  Serial.print("-");  if (x < 1000) Serial.print(" ") ;  if (x < 100) Serial.print(" ") ;  Serial.print(x, DEC);  } else { // odd  Serial.print(" ");  Serial.print("+");  if (x < 1000) Serial.print(" ") ;  if (x < 100) Serial.print(" ") ;  Serial.print(x, DEC);  if (i < results->rawlen - 1) Serial.print(", "); //',' not needed for last one  }  if (!(i % 8)) Serial.println("");  }  Serial.println(""); // Newline }  //+============================================================================= // Dump out the decode\_results structure. // void dumpCode (decode\_results \*results) {  // Start declaration  Serial.print("unsigned int "); // variable type  Serial.print("rawData["); // array name  Serial.print(results->rawlen - 1, DEC); // array size  Serial.print("] = {"); // Start declaration   // Dump data  for (int i = 1; i < results->rawlen; i++) {  Serial.print(results->rawbuf[i] \* USECPERTICK, DEC);  if ( i < results->rawlen - 1 ) Serial.print(","); // ',' not needed on last one  if (!(i & 1)) Serial.print(" ");  }   // End declaration  Serial.print("};"); //   // Comment  Serial.print(" // ");  encoding(results);  Serial.print(" ");  ircode(results);   // Newline  Serial.println("");   // Now dump "known" codes  if (results->decode\_type != UNKNOWN) {   // Some protocols have an address  if (results->decode\_type == PANASONIC) {  Serial.print("unsigned int addr = 0x");  Serial.print(results->address, HEX);  Serial.println(";");  }   // All protocols have data  Serial.print("unsigned int data = 0x");  Serial.print(results->value, HEX);  Serial.println(";");  } }  //+============================================================================= // The repeating section of the code // void loop ( ) {  decode\_results results; // Somewhere to store the results   if (irrecv.decode(&results)) { // Grab an IR code  dumpInfo(&results); // Output the results  dumpRaw(&results); // Output the results in RAW format  dumpCode(&results); // Output the results as source code  Serial.println(""); // Blank line between entries  irrecv.resume(); // Prepare for the next value  } } |