ENVIRONMENTAL MONITORING

Creating a environmental monitoring system framework uses sensors for encompassing area moistness and temperature. these information could be used to animate transient conduct like gadget becoming hot or getting cool down and other long haul insight of the gadgets.

HARDWARE REQUIREMENTS

- **PROCESSOR:I3 OR ADVANCED**
- **RAM:4GB**
- **HARD DISK DRIVE:256GB**
- **USB TYPE A,B**
- **ARDUINO BOARD: UNO R4 WIFI**
- **MOUSE,KEYBORD AND OTHER SENSORS**

CODING

```
#include <SDI12.h>

 #define SERIAL_BAUD 115200 /*!< The baud rate for the output serial port */

#define DATA_PIN 7 /*!< The pin of the SDI-12 data bus */

#define POWER_PIN 22 /*!< The sensor power pin (or -1 if not switching power) */

/** Define the SDI-12 bus */

SDI12 mySDI12(DATA_PIN);

// keeps track of active addresses

bool isActive[64] = {

0,

};

uint8_t numSensors = 0;

/**

* @brief converts allowable address characters ('0'-'9', 'a'-'z', 'A'-'Z') to a

* decimal number between 0 and 61 (inclusive) to cover the 62 possible

* addresses.

 */

byte charToDec(char i) {
```

```
if ((i >= '0') && (i <= '9')) return i - '0';
if ((i >= 'a') && (i <= 'z')) return i - 'a' + 10;
if ((i >= 'A') && (i <= 'Z'))
return i - 'A' + 36;
else
return i;
}
/**
* @brief maps a decimal number between 0 and 61 (inclusive) to allowable
* address characters '0'-'9', 'a'-'z', 'A'-'Z', *
* THIS METHOD IS UNUSED IN THIS EXAMPLE, BUT IT MAY BE HELPFUL. */
char decToChar(byte i) {
if (i < 10) return i + '0';
if ((i >= 10) && (i < 36)) return i + 'a' - 10;
if ((i >= 36) && (i <= 62))
return i + 'A' - 36;
else
return i;
}
/**
* @brief gets identification information from a sensor, and prints it to the serial
* port
 *
* @param i a character between '0'-'9', 'a'-'z', or 'A'-'Z'.
*/
void printInfo(char i) {
String command = "";
command += (char)i;
command += "I!";
mySDI12.sendCommand(command);
delay(100);
```

```
String sdiResponse = mySDI12.readStringUntil('\n');
sdiResponse.trim();
// allccccccccmmmmmmmvvvxxx...xx<CR><LF>
Serial.print(sdiResponse.substring(0, 1)); // address
Serial.print(", ");
Serial.print(sdiResponse.substring(1, 3).toFloat() / 10); // SDI-12 version number
Serial.print(", ");
Serial.print(sdiResponse.substring(3, 11)); // vendor id
Serial.print(", ");
Serial.print(sdiResponse.substring(11, 17)); // sensor model
Serial.print(", ");
Serial.print(sdiResponse.substring(17, 20)); // sensor version
Serial.print(", ");
Serial.print(sdiResponse.substring(20)); // sensor id
Serial.print(", ");
}
bool getContinuousResults(char i, int resultsExpected) {
uint8_t resultsReceived = 0;
uint8_t cmd_number = 0;
while (resultsReceived < resultsExpected && cmd_number <= 9) {
String command = "";
// in this example we will only take the 'DO' measurement
command = "";
command += i;
command += "R";
command += cmd_number;
command += "!"; // SDI-12 command to get data [address][D][dataOption][!]
mySDI12.sendCommand(command);
uint32_t start = millis();
while (mySDI12.available() < 3 && (millis() - start) < 1500) {}
mySDI12.read(); // ignore the repeated SDI12 address
```

```
char c = mySDI12.peek(); // check if there's a '+' and toss if so
if (c == '+') { mySDI12.read(); }
while (mySDI12.available()) {
char c = mySDI12.peek();
if (c == '-' || (c >= '0' && c <= '9') || c == '.') {
float result = mySDI12.parseFloat(SKIP_NONE);
Serial.print(String(result, 10));
if (result != -9999) { resultsReceived++; }
} else if (c == '+') {
mySDI12.read();
Serial.print(", ");
} else {
mySDI12.read();
}
delay(10); // 1 character ~ 7.5ms
}
if (resultsReceived < resultsExpected) { Serial.print(", "); }
cmd_number++;
}
mySDI12.clearBuffer();
return resultsReceived == resultsExpected;
}
// this checks for activity at a particular address
// expects a char, '0'-'9', 'a'-'z', or 'A'-'Z' boolean checkActive(char i) {
String myCommand = "";
myCommand = "";
myCommand += (char)i; // sends basic 'acknowledge' command [address][!]
myCommand += "!";
for (int j = 0; j < 3; j++) { // goes through three rapid contact attempts
mySDI12.sendCommand(myCommand);
delay(100);
```

```
if (mySDI12.available()) { // If we here anything, assume we have an active sensor
mySDI12.clearBuffer();
return true;
}
}
mySDI12.clearBuffer();
return false;
}
void setup() {
Serial.begin(SERIAL_BAUD);
while (!Serial)
;
Serial.println("Opening SDI-12 bus...");
mySDI12.begin();
delay(500); // allow things to settle
Serial.println("Timeout value: ");
Serial.println(mySDI12.TIMEOUT);
// Power the sensors;
if (POWER_PIN > 0) {
Serial.println("Powering up sensors...");
pinMode(POWER_PIN, OUTPUT);
digitalWrite(POWER_PIN, HIGH);
delay(200);
}
// Quickly Scan the Address Space
Serial.println("Scanning all addresses, please wait...");
Serial.println("Sensor Address, Protocol Version, Sensor Vendor, Sensor Model, "
"Sensor Version, Sensor ID");
for (byte i = 0; i < 62; i++) {
char addr = decToChar(i);
if (checkActive(addr)) {
```

```
numSensors++;
isActive[i] = 1;
printInfo(addr);
Serial.println();
}
}
Serial.print("Total number of sensors found: ");
Serial.println(numSensors);
if (numSensors == 0) {
Serial.println(
"No sensors found, please check connections and restart the Arduino.");
while (true) { delay(10); } // do nothing forever
}
Serial.println();
Serial.println( "Time Elapsed (s), Sensor Address, Est Measurement Time (s), Number Measurements,
"
"Real Measurement Time (ms), Measurement 1, Measurement 2, ... etc.");
Serial.println(
 "--------------------------------------------------------------------------------");
}
void loop() {
// measure one at a time
for (byte i = 0; i < 62; i++) {
char addr = decToChar(i);
if (isActive[i]) {
// Serial.print(millis() / 1000);
Serial.print(millis());
Serial.print(", ");
getContinuousResults(addr, 4);
Serial.println();
}
```

```
}
delay(5000L); // wait ten seconds between measurement attempts.
}
```