Commits

petewill committed `6d9e81f`
3 minutes ago

Approve

Pete's edits.
- Added MySensors info at top of code
- Updated LCD code to work with new Arduino IDE version
- Added missing button code
- Added code to update valve times on initial power up.
- Updated code to stop irrigation if running and button is pressed

`57f1fc0`
View raw commit
Stop watching

Comments (0)

What do you want to say?

Files changed (1)

+79 -23 IrrigationController/IrrigationController.ino

IrrigationController/IrrigationController.ino — Side-by-side diff | View file | Comment

```diff
 #include <SPI.h>
 #include <LiquidCrystal_I2C.h>
 //
-#define NUMBER_OF_VALVES 8  // Change this to set your valve count up to 16.
+#define NUMBER_OF_VALVES 5  // Change this to set your valve count up to 16.
 #define VALVE_RESET_TIME 7500UL   // Change this (in milliseconds) for the time you need your valves to hydraulically res
-#define RADIO_ID 9 // AUTO  // Change this to fix your Radio ID or use Auto
+#define RADIO_ID AUTO // AUTO  // Change this to fix your Radio ID or use Auto
+#define SKETCH_NAME "MySprinkler"
+#define SKETCH_VERSION "2.0"
+//
+#define CHILD_ID_SPRINKLER 0
 //
 #define ACTIVE_LOW // comment out this line if your relays are active high
 //
```

```diff
 int lastValve;
 unsigned long startMillis;
 const int ledPin = 5;
+const int waterButtonPin = 3;
 boolean buttonPushed = false;
 boolean showTime = true;
 boolean clockUpdating = false;
```

```diff
 const int clockPin = 4;
 const int dataPin  = 7;
 //
-uint8_t clock[8] = {0x0,0xe,0x15,0x17,0x11,0xe,0x0}; // fetching time indicator
-uint8_t raindrop[8] = {0x4,0x4,0xA,0xA,0x11,0xE,0x0,}; // fetching Valve Data indicator
-//
-LiquidCrystal_I2C lcd(0x27, 16, 2);  // set the LCD I2C address to 0x27 (16 characters and 2 line display)
+byte clock[8] = {0x0,0xe,0x15,0x17,0x11,0xe,0x0}; // fetching time indicator
+byte raindrop[8] = {0x4,0x4,0xA,0xA,0x11,0xE,0x0,}; // fetching Valve Data indicator
+LiquidCrystal_I2C lcd(0x27);  // set the LCD I2C address to 0x27
 MySensor gw;
 //
-MyMessage msg1valve(0,V_LIGHT);
-MyMessage var1valve(0,V_VAR1);
-MyMessage var2valve(0,V_VAR2);
+MyMessage msg1valve(CHILD_ID_SPRINKLER,V_LIGHT);
+MyMessage var1valve(CHILD_ID_SPRINKLER,V_VAR1);
+MyMessage var2valve(CHILD_ID_SPRINKLER,V_VAR2);
 //
 void setup()
 {
- Serial.begin(115200);
- delay(5000);
- lcd.init();
+  delay(5000);
+ //*Added
+ lcd.begin(16, 2); //(16 characters and 2 line display)
+//*Removed
+// lcd.init();
  lcd.clear();
```

```
129 135  lcd.backlight();
130 136  lcd.createChar(0, clock);
```

...

```
134 140  pinMode(clockPin, OUTPUT);
135 141  pinMode(dataPin, OUTPUT);
136 142  pinMode(ledPin, OUTPUT);
137      - attachInterrupt(2, PushButton, CHANGE);
     143 + pinMode(waterButtonPin, INPUT_PULLUP);
     144 + attachInterrupt(1, PushButton, RISING); //May need to change for your Arduino model
138 145  digitalWrite (ledPin, HIGH);
139 146  //
140 147  //check for saved date in EEPROM
```

...

```
149 156     }
150 157  }
151 158  gw.begin(getVariables, RADIO_ID, false); // Change 'false' to 'true' to create a Radio repeating node
152      - gw.sendSketchInfo("MySprinkler", "2.0");
     159 + gw.sendSketchInfo(SKETCH_NAME, SKETCH_VERSION);
153 160  for (byte i = 0; i <= NUMBER_OF_VALVES; i++)
154 161  {
155 162     gw.present(i, S_LIGHT);
```

...

```
164 171  lcd.setCursor(0, 0);
165 172  lcd.print(F(" Syncing Time  "));
166 173  lcd.setCursor(15, 0);
167      - lcd.write(0);  //lcd.print(0, BYTE);
     174 + lcd.write(byte(0));  //lcd.print(0, BYTE);
168 175  lcd.setCursor(0, 1);
169 176  int clockCounter = 0;
170 177  while(timeStatus() == timeNotSet && clockCounter < 21)
```

...

```
174 181     DEBUG_PRINTLN(F("Requesting time from Gateway:"));
175 182     delay(1000);
176 183     lcd.print(".");
177      -    DEBUG_PRINT(F("."));
     184 +    //DEBUG_PRINT(F("."));
178 185     clockCounter++;
179 186     if (clockCounter > 16)
180 187     {
```

...

```
189 196  }
190 197  //
191 198  lcd.clear();
     199 + //Update valve times when first powered on
     200 + for(byte i = 0; i <= NUMBER_OF_VALVES; i++)
     201 + {
     202 +   allZoneTime[i]=-1;
     203 +   valveSoloTime[i]=-1;
     204 +   int clockCounter = 0;
     205 +   while((valveSoloTime[i]==-1 || allZoneTime[i]==-1) && clockCounter < 10)
     206 +   {
     207 +     DEBUG_PRINTLN(F("Calling for Valve Times..."));
     208 +     lcd.setCursor(15, 0);
     209 +     lcd.write(byte(1)); //lcd.write(1);
     210 +     gw.request(i, V_VAR1);
     211 +     gw.request(i, V_VAR2);
     212 +     delay(1000);
     213 +     gw.process();
     214 +     clockCounter++;
     215 +     if (clockCounter > 10)
     216 +     {
     217 +        DEBUG_PRINTLN(F("Failed initial valve synchronization!"));
     218 +        lcd.clear();
     219 +        lcd.print(F("  Failed Valve  "));
     220 +        lcd.setCursor(0,1);
     221 +        lcd.print(F(" Syncronization "));
     222 +     }
     223 +   }
     224 + }
     225 +
192 226 }
193 227 //
194 228 void loop()
```

...

```
201 235  if (buttonPushed)
202 236  {
203 237     DEBUG_PRINTLN(F("Button Pressed"));
204      -    if (state != RUN_ALL_ZONES);
     238 +    if (state != RUN_ALL_ZONES)
205 239     {
206 240      state = RUN_ALL_ZONES;
207 241      valveNumber = 1;
```

...

```
223 257      DEBUG_PRINT(F("State = "));
```

```
224 258      DEBUG_PRINTLN(state);
225 259     }
    260 +   else
    261 +   {
    262 +     state = STAND_BY_ALL_OFF;
    263 +     //valveNumber = 1;
    264 +     gw.send(msg1valve.setSensor(0).set(false), false);
    265 +     startMillis = millis();
    266 +     for (byte i = 0; i < 5; i++)  // flash lcd backlight on button press
    267 +     {
    268 +       lcd.noBacklight();
    269 +       delay(25);
    270 +       lcd.backlight();
    271 +     }
    272 +     delay(50);
    273 +     fastClear();
    274 +     lcd.setCursor(0,0);
    275 +     lcd.print(F("*AllZone Off*"));
    276 +     lcd.setCursor(0,0);
    277 +     lcd.print(F(" Stopping  Zones "));
    278 +     delay(1000);
    279 +     DEBUG_PRINT(F("State = "));
    280 +     DEBUG_PRINTLN(state);
    281 +   }
226 282     buttonPushed = false;
227 283   }
228 284   if (state == STAND_BY_ALL_OFF)
```

...

```
502 558         lcd.print(F("  System Ready "));
503 559         if (clockUpdating)
504 560         {
505     -        lcd.setCursor(15,0);
506     -        lcd.write(0);
    561 +        lcd.setCursor(15,0);
    562 +        lcd.write(byte(0));
507 563         }
508 564         lcd.setCursor(0,1);
509 565         lcd.print(hourFormat12() < 10 ? F(" ") : F(""));
```

...

```
525 581         if (clockUpdating)
526 582         {
527 583          lcd.setCursor(15,0);
528     -        lcd.write(0);
    584 +        lcd.write(byte(0));
529 585         }
530 586         lcd.setCursor(0,1);
531 587         lcd.print(dayOfWeek[weekday(lastTimeRun)]);
```

...

```
657 713 void updateClock()
658 714 {
659 715   static unsigned long lastVeraGetTime;
660     - if (millis() - lastVeraGetTime >= 60000UL) // updates clock time and gets zone times from vera once every 10 minutes
    716 + if (millis() - lastVeraGetTime >= 3600000UL) // updates clock time and gets zone times from vera once every hour
661 717   {
662 718     DEBUG_PRINTLN(F("Requesting time and valve data from Gateway...")); 
663 719     lcd.setCursor(15,0);
664     -   lcd.write(0);
    720 +   lcd.write(byte(0));
665 721     clockUpdating = true;
666 722     gw.requestTime(receiveTime);
667 723     lastVeraGetTime = millis();
```

...

```
690 746   {
691 747     DEBUG_PRINTLN(F("Calling for Valve Times..."));
692 748     lcd.setCursor(15, 0);
693     -   lcd.write(1);
    749 +   lcd.write(byte(1)); //lcd.write(1);
694 750     gw.request(valveIndex, V_VAR1);
695 751     gw.request(valveIndex, V_VAR2);
696 752     valveUpdateTime = millis();
```

...