Vending Machine MK I

January 12th, 2026

Designed and manufactured by Herr Technik

Follow me on my social media to receive the latest updates of my projects

Youtube Channel: HerrTechnik

https://www.youtube.com/@HerrTechnikMKI

E-mail for business and sponsorships:

contactotransistorizedmx@gmail.com

Donations:

https://buymeacoffee.com/herrtechnik

https://paypal.me/HerrTechnik

Description

This is the first machine of a series of open-source vending machines. The primary goal of this machine was to minimize the tradeoff between the price of the final product and its reliability in the real world. This machine uses the industrial-grade PIC16F887 microcontroller, which is known for its robustness and low price. It also features an almost barebones design and reduced BOM (Bill of Materials) that keeps the machine operative while avoiding wasteful components. This machine can sell up to ten different products, but the code and hardware can be adapted for more or fewer products. In this version, the pumps are activated using an open loop control, only relying on the timing configured by the user to accurately fill the product. It would be a better option to use flow sensors, however, this action will raise the cost excessively and for the purposes of this first project, it is acceptable not to include a close loop control. This machine is ready to be used and does not require any additional code or hardware to operate normally.

In future versions, the STM32 microcontroller will be used to include more sophisticated features such as modern payment methods, speakers, and IoT monitoring. These newer versions will also use stainless steel instead of galvanized steel and they are going to be optimized for mass production with CAM simulations and indeed an industrial-grade PCB.

WARNINGS:

- The 74LS14 (Hex inverting Schmitt-trigger) chip also needs a decoupling 100 nF ceramic capacitor.
- Unused inputs of the 74LS14 should be connected to ground and the outputs should not be connected to anything.
- If you are using an active low relay board then GND is connected to the relays, so this pin MUST NOT be connected to the thin GND wires used for logic signals, instead by a thick cable and the same for JD_VDD. The Vcc pin is just used to turn on the LED of each optocoupler, so it's reasonable to use a thinner wire.
- It is absolutely important to filter the output provided by the coin acceptor via the proposed circuit, otherwise the credit counter might increase erratically
- Make sure to use properly sized wires and cables for each component

DESIGN DECISIONS I MADE:

- The PIC16F887 was selected because it has enough computing power to operate the machine correctly, it can retain programmable values for a very long time using its EEPROM and it's very reliable. Other microcontrollers can be used, but preferably one from the STM32 family.

- An i2c LCD screen module was used instead of a normal module to save pins for future hardware upgrades that might need it.
- 3.3k pull-up resistors where used for the i2c display module to favor high-speed communications, the same happened with the button matrix pull-up resistors connected to the rows since their state is constantly being changed. To lower power-consumption the pull-down resistors used for the special button and the mode switch got a higher value since they are almost never used.
- Two 74LS595 IC's (8-bit shift register) were used despite there were enough pins to drive the two relay modules to save pins for future hardware upgrades and to manage other charges such as coin-hoppers, LED-illumination, alarms, etc.
- The OE (Output Enable) pin of both 74LS595's should be held high (high impedance) on start by the 6.8k resistor to avoid turning on all relays of the module and then it should be set low (activate its outputs) only after both 74LS595's have been properly initialized.
- I recommend using a fuse for each pump, a fuse for the control board, and a fuse for the entire system. I used only one fuse due to hard cost-constraints, but future versions will use the fuses needed.
- The reason why I used two 74LS93 IC's (4-bit counters) to count the pulses instead of just using an interruption is because the LCD library (AFAIK) overrides the use of interruptions as it was tested several times. So, one feasible way was to use these counters, and they would just wait for the PIC to read their value and then, they would be reset.

Category	Name	Units	Price	Total cost
Electronics	PIC16F887 Microcontroller	1	$ 124.60	$ 124.60
Electronics	40-pin integrated circuit base DIP	1	$ 6.00	$ 6.00
Electronics	74HC595 shift register	2	$ 10.00	$ 20.00
Electronics	74LS93 binary counter	2	$ 20.00	$ 40.00
Electronics	74LS14 schmitt trigger	1	$ 25.00	$ 25.00
Electronics	4N25 optocoupler	1	$ 9.00	$ 9.00
Electronics	100 nF ceramic capacitor	5	$ 3.00	$ 15.00
Electronics	1000 uF Electrolytic capacitor 25 V	1	$ 7.00	$ 7.00
Electronics	22 AWG wires for perfboard connections	4.5	$ 5.00	$ 22.50
Electronics	22 AWG duplex cables for buttons	5	$ 5.00	$ 25.00
Electronics	18 gauge duplex cable for pumps	7.22	$ 13.00	$ 93.86
Electronics	14 gauge duplex cable	4.4	$ 22.50	$ 99.00
Electronics	Power supply CA to 12 V 12.5 A	1	$ 225.22	$ 225.22
Electronics	7805 regulator	1	$ 9.00	$ 9.00
Electronics	TO-220 Heatsink	1	$ 11.00	$ 11.00
Electronics	Perforated phenolic board 107 mm x 140 mm	1	$ 79.00	$ 79.00
Electronics	Tin solder	2	$ 39.00	$ 78.00
Electronics	6.8k resistor	2	$ 1.00	$ 2.00
Electronics	1k resistor	1	$ 1.00	$ 1.00
Electronics	3.3k resistor	5	$ 1.00	$ 5.00
Electronics	390 resistor	1	$ 1.00	$ 1.00
Electronics	330 resistor	1	$ 1.00	$ 1.00
Electronics	Green LED	1	$ 2.00	$ 2.00
Electronics	8-relay module 12 V	2	$ 83.89	$ 167.77
Electronics	LCD Screen 1602 with i2c module	1	$ 79.00	$ 79.00
Electronics	SR 500 Coin acceptor	1	$ 469.00	$ 469.00
Electronics	Diaphragm pumps 12 V 70 W	10	$ 163.86	$ 1,638.60
Electronics	11 arcade buttons + 1 replacement	1	$ 107.10	$ 107.10
Electronics	DIP switch x2	1	$ 4.00	$ 4.00
Electronics	Two-row terminal bank, 24 screws	0.5	$ 39.00	$ 19.50

Electronics	1/4 brass faston terminal	22	$ 2.00	$ 44.00
Electronics	2x screw terminal	13	$ 5.00	$ 65.00
Electronics	American fuse holder	1	$ 19.00	$ 19.00
Electronics	10 Amp american fuse	1	$ 20.00	$ 20.00
Electronics	Grounded plastic industrial plug	1	$ 18.00	$ 18.00
Electronics	Kit termofit	0.18	$ 59.00	$ 10.62
Electronics	Bag with 50 cable ties	0.36	$ 20.00	$ 7.20
Electronics	4.5 mm MDF panel for electric panel	0.0288	$ 240.00	$ 6.92
Electronics	Wooden screws	11	$ 0.50	$ 5.50
Hydraulics	1/2 plastic hose	9	$ 15.00	$ 135.00
Hydraulics	10-piece stainless steel hose clamps set	2	$ 48.00	$ 96.00
Hydraulics	1/2 PVC elbow 45°	10	$ 2.60	$ 26.00
Hydraulics	1/2 x 3/8 PVC externally-threaded insertion adapter (truper 48573)	10	$ 4.20	$ 42.00
Hydraulics	1/2 PVC female adapter (truper 45432)	10	$ 3.30	$ 33.00
Hydraulics	1/2 " PVC pipe	0.1	$ 43.00	$ 4.30
Hydraulics	1/8 Female hose adapter	20	$ 6.00	$ 120.00
Painting	Paint (the one that you consider the best)	0.6	$ 195.00	$ 117.00
Painting	Half a liter of thinner	1	$ 50.00	$ 50.00
Welding	18 gauge galvanized steel sheet 3 ft x 10 ft	0.3226	$ 873.00	$ 281.60
Welding	14 gauge PTR 1 x 1	0.9786	$ 218.00	$ 213.34
Welding	1/8 x 3/4 Flat bar	0.5278	$ 81.60	$ 43.07
Welding	3/8 Square bar	0.5961	$ 113.12	$ 67.43
Welding	3/16 x 1/4 rivets	1.8	$ 26.00	$ 46.80
Welding	1 kg of 1/8 E6013 electrodes	1	$ 89.00	$ 89.00
Welding	Flap disc for angle grinder	1	$ 37.00	$ 37.00
Welding	Cutting disc for angle grinder	1	$ 15.50	$ 15.50
Welding	1 meter of 3/8 threaded rod for anchors	1	$ 46.00	$ 46.00
Welding	3/8 nuts for anchors	4	$ 1.93	$ 7.72
Welding	1/8 x 1.5" bolts	8	$ 0.60	$ 4.80
Welding	1/8 nuts	8	$ 0.25	$ 2.00
Welding	3/16 x 2" bolts	30	$ 1.00	$ 30.00

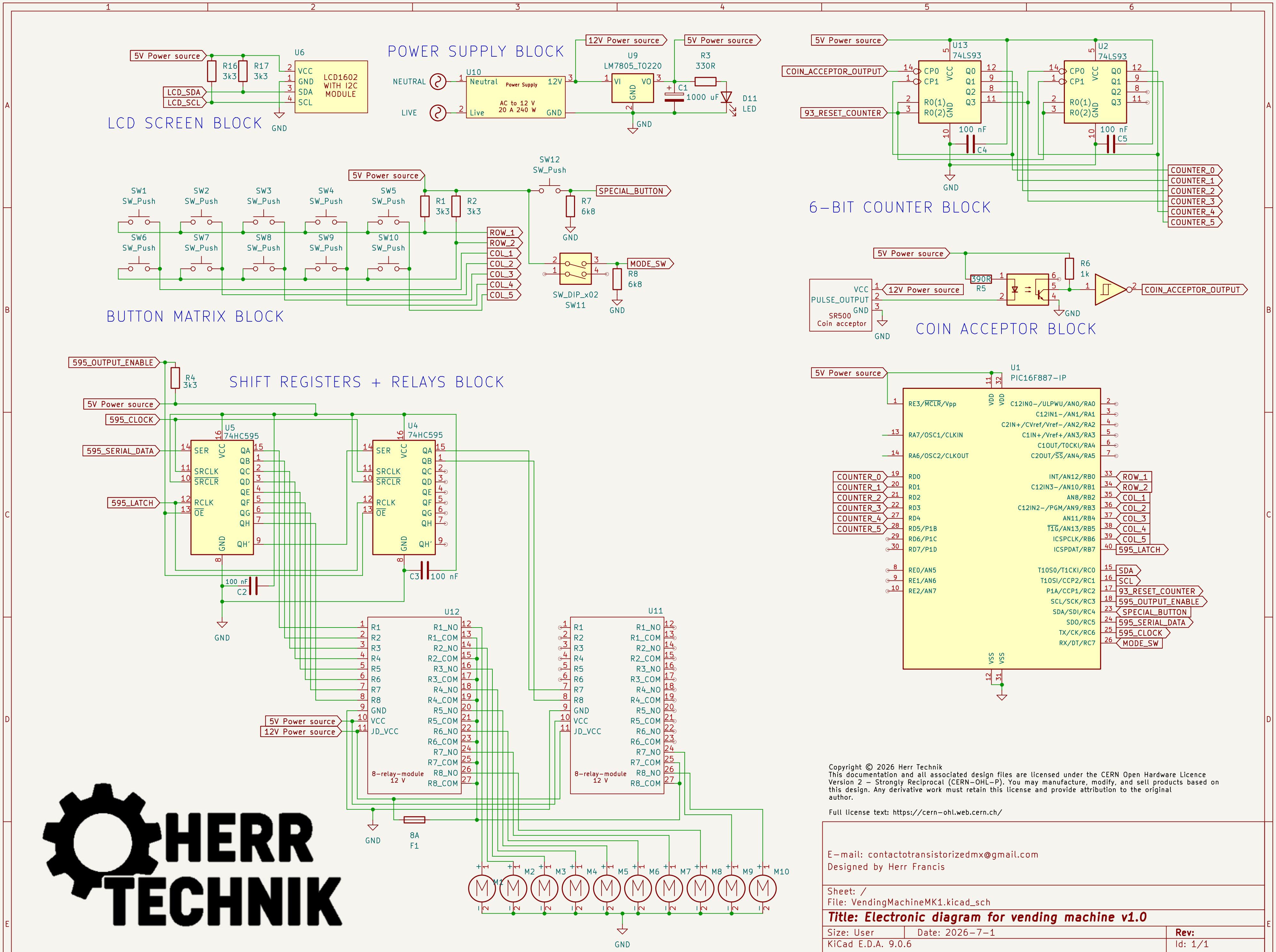
Welding	3/16 nut	30	$ 1.00	$ 30.00
Welding	3/8 Cylindrical hinges	4	$ 7.00	$ 28.00
			TOTAL COST	$ 5,147.94
			IN USD (18 PESOS = 1 DOLLAR)	$ 286.00

```
/*
   Copyright (c) 2026 Herr Technik
   This program is free software: you can redistribute it and/or modify it under the terms of
   the GNU General Public License as published by the Free Software Foundation, either
   version 3 of the License, or (at your option) any later version.
   This program is distributed in the hope that it will be useful, but WITHOUT ANY
   WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
   FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
   You should have received a copy of the GNU General Public License along with this
   program. If not, see https://www.gnu.org/licenses/.

   This code assumes the following:
      > You are using an active low relay board (activatePump function sends
         a zero to activate the corresponding pump, and 1's to deactivate the others)

   Follow me on my social media to receive the latest updates of my projects
      Youtube Channel: HerrTechnik
      https://www.youtube.com/@HerrTechnikMKI
   E-mail for business and sponsorships:
      contactotransistorizedmx@gmail.com
   Donations:
      https://buymeacoffee.com/herrtechnik
      https://paypal.me/HerrTechnik
*/

#include <16F887.h>
#fuses INTRC_IO, NOWDT, NOPROTECT, BROWNOUT
#use delay(clock=4000000)
#use I2C(MASTER, SDA=PIN_C0, SCL=PIN_C1, FAST)
#define ADDRESS_LCD 0x4E
#include <i2c_Flex_LCD.c> // LCD library
#include <stdint.h> // integers library

// SETTINGS
#define ENGLISH // Define this if you want English, if not defined, Spanish will be used
#define STD_DELAY 1500// Standard delay
#define TINY_DELAY 50 // Tiny delay
#define PRODUCTS_NUMBER 10 // Number of products (varies according to the model)
#define ROWS_NUMBER 2 // Number of rows in the button matrix
#define COLUMNS_NUMBER 5 // Number of columns in the button matrix
#define LCD_DISPLAY_ADDRESS 0x4E // i2c address used to communicate with LCD display
#define LCD_DISPLAY_COLUMNS 16 // Number of columns in the LCD display
#define LCD_DISPLAY_ROWS 2 // Number of rows in the LCD display
#define NULL_CHAR 0x00 // Null character that is interpreted as "No button pressed"
// This value must be entered to activatePump() function to disable all operative pumps
#define DISABLE_PUMPS 16
```

```c
#define SHIFT_REGISTER_OUTPUTS 16 // Number of outputs when the two 595 shift registers are combined

// MACROS
#define VALUE_IS_NOT_A_PRODUCT(k) ((k) < 'A' || (k) > 'J') // Opposite of VALUE_IS_A_PRODUCT
// This macro checks if the value is between A and J inclusive
#define VALUE_IS_A_PRODUCT(k) ((k) >= 'A' && (k) <= 'J')
#define ASCII_TO_INDEX(k) ((k) - 'A') // This macro converts an ascii character into a zero-based index

// Define the names of the products
#ifdef ENGLISH
   #define PRODUCT1 "Lemon"
   #define PRODUCT2 "Grapes"
   #define PRODUCT3 "Orange"
   #define PRODUCT4 "Soda"
   #define PRODUCT5 "Beer"
   #define PRODUCT6 "Vodka"
   #define PRODUCT7 "Whisky"
   #define PRODUCT8 "Milk"
   #define PRODUCT9 "Water"
   #define PRODUCT10 "Pancho cola"
#else
   #define PRODUCT1 "Cloro"
   #define PRODUCT2 "Pinol verde"
   #define PRODUCT3 "Lavanda"
   #define PRODUCT4 "Citrico"
   #define PRODUCT5 "Pera manzana"
   #define PRODUCT6 "Mas color"
   #define PRODUCT7 "Ariel"
   #define PRODUCT8 "Flor de luna"
   #define PRODUCT9 "Axion"
   #define PRODUCT10 "Desengrasante"
#endif

// Pins definitions
   // Pins connected to the 2x5 button matrix
   #define ROW1 PIN_B0// Inputs They need weak pull up resistors (around 10k)
   #define ROW2 PIN_B1
   #define COL1 PIN_B2 // OUTPUTS
   #define COL2 PIN_B3
   #define COL3 PIN_B4
   #define COL4 PIN_B5
   #define COL5 PIN_B6
   const int16 ROWS[ROWS_NUMBER] = {ROW1, ROW2};
   const int16 COLUMNS[COLUMNS_NUMBER] = {COL1, COL2, COL3, COL4, COL5};
   // Pins connected to 595 IC
   #define SER PIN_C5 // Pin that sends serial data to 595 IC
   #define LATCH PIN_B7 // Pin that enables internal register of 595 IC
   #define CLK PIN_C6 // Pin that sends clock pulses to shift register of 595 IC
   // Pin that pulls low the output enable pin of both 595's
   // after it's been charged with ones (to avoid burnign the fuse at start-up)
   #define RELAY_ENABLE PIN_C3
   // Pins connected to the two 7493 binary counters
   // Output used to reset both counters when the IC successfully reads the stored value in them
   #define COUNTER_RESET PIN_C2
   #define COUNTER0 PIN_D0 // Inputs used to receive what is stored in the counters
   #define COUNTER1 PIN_D1
   #define COUNTER2 PIN_D2
   #define COUNTER3 PIN_D3
   #define COUNTER4 PIN_D4
   #define COUNTER5 PIN_D5 // Up to number 63
   // Pins connected to the especial function button and the mode switch
   #define MODE_SW PIN_C7 // This is connected to the dip switch (pull-down resistor)
   #define SPECIAL_BUTTON PIN_C4 // This is connected to an arcade button (pull-down resistor)

// Prototypes for custom functions
   int8 scanKeypad(void); // Basic function used to analyze the keypad
   void keypad(void); // Advanced function used to keep track of previously pressed keys and decrease bouncing
   void activatePump(int val); // Function used to activate the required pump, only one at a time
   void stdDelay(void); // Function that contains the standard delay
   void tinyDelay(void); // Function that contains the tiny delay
   void printStringROM(int16 ptr); // This function prints a string from the ROM memory
   void clearScreen(void); // Clear screen
   void showNames(int16 namePtr); // Clears the screen and shows tha name of the selected product
   // Displays the required message to the user
   void showMessage(uint8_t message, int1 delay = 1, int1 clear = 1);

// EEPROM ADDRESSES
#define START_ADDRESS 0x00 // Address at which EEPROM use starts
#define PRICE_ADDRESS START_ADDRESS // 10 consecutive bytes to store the price of each product
// 10 consecutive bytes to store the sales of each product
#define SALES_ADDRESS PRICE_ADDRESS + PRODUCTS_NUMBER
// 10 consecutive bytes to store the availability of each product
#define AVAILABILITY_ADDRESS SALES_ADDRESS + PRODUCTS_NUMBER
// 10 consecutive bytes to store the time it takes for each corresponding pump
// to dispense one liter of that product 1 means 100 ms and 255 means 25.5 seconds
#define TIMER_ADDRESS AVAILABILITY_ADDRESS + PRODUCTS_NUMBER
// 4 consecutive bytes to store the total sales as a 4-byte unsigned integer
#define TOTAL_SALES_ADDRESS TIMER_ADDRESS + PRODUCTS_NUMBER

// Global variables
   // Variables used for display
   uint8_t key = NULL_CHAR;
   int16 productNames[PRODUCTS_NUMBER];
   // Variables used for money
      // This array will be populated with the eeprom data (0x00 to 0x09)
      int8 productPrice[PRODUCTS_NUMBER];
      // Same but (0x14 to 0x1D) records whether a product is available or not
      int8 productAvailable[PRODUCTS_NUMBER];
      // This array will store the time it takes for a pump to dispense 1 L of the corresponding product
      int8 productTimer[PRODUCTS_NUMBER];
      // Variables used for processing
      uint8_t selectedProduct; // Variable used to store the selected product
      volatile unsigned int8 credit = 0; // Variable used to store the current credit of the user
      // This variable is used to stabilize the reading process of the two binary counters and avoid errors
      volatile unsigned int8 previousCredit = 0;
      // Boolean variable used as a flag to indicate if the user has inserted money.
      volatile int1 moneyInserted = false;

   // Configuration variables
   int8 matrix[ROWS_NUMBER][COLUMNS_NUMBER] = // rows x columns
   {
      {'J', 'H', 'F', 'D', 'B'},
      {'I', 'G', 'E', 'C', 'A'}
   };

// Function used to read the value stored in the two 7493 counters and add it to the credit variable
// Interrupt Service Routine
#INT_TIMER1
void readCounter(void)
{
   if (input_d() != 0x00)
   {
      if (previousCredit == input_d())
      {
         credit += input_d() & 0b00111111; // And add that to the credit variable
         output_high(COUNTER_RESET); // Reset both counters
         output_low(COUNTER_RESET);
         moneyInserted = true;
      }
      else
      {
         previousCredit = input_d();
      }
   }
}

void main(void)
{
   // Configuration
      // PORT A (UNUSED)
      set_tris_a(0b00000000);
      output_a(0b00000000);
      // PORT B(KEYPAD)
      set_tris_b(0b00000011); // Keypad's pins | rows are inputs and columns are outputs
      output_b(0b00000000); // MSB - LSB
      for (int8 i = 0; i < COLUMNS_NUMBER; i++)
      {
         output_float(COLUMNS[i]);
      }
      // PORT C (I2C, 595 AND 7493 CONTROL
      set_tris_c(0b00000000);
      output_c(0b00001000); // MSB - LSB
      // PORT D (INPUT FROM COUNTERS)
      set_tris_d(0b00111111);
      output_d(0b00000000); // MSB - LSB
      // Setup timer 1
      setup_timer_1(T1_INTERNAL | T1_DIV_BY_8);
      disable_interrupts(GLOBAL);
      // LCD
      lcd_init(LCD_DISPLAY_ADDRESS, LCD_DISPLAY_COLUMNS, LCD_DISPLAY_ROWS);
      // Initialize productNames array
      showMessage(0, 1, 1); // "Starting machine"
      productNames[0] = &PRODUCT1;
      productNames[1] = &PRODUCT2;
      productNames[2] = &PRODUCT3;
      productNames[3] = &PRODUCT4;
      productNames[4] = &PRODUCT5;
      productNames[5] = &PRODUCT6;
      productNames[6] = &PRODUCT7;
      productNames[7] = &PRODUCT8;
      productNames[8] = &PRODUCT9;
      productNames[9] = &PRODUCT10;

      // Populating product availability array, product price array and product timer array
      for (int8 i = 0; i < PRODUCTS_NUMBER; i++)
      {
         productPrice[i] = read_eeprom(PRICE_ADDRESS + i);
         productAvailable[i] = read_eeprom(AVAILABILITY_ADDRESS + i);
         productTimer[i] = read_eeprom(TIMER_ADDRESS + i);
      }
      output_high(COUNTER_RESET); // Reset both counters
      output_low(COUNTER_RESET);

      // Deactivate all pumps and enable relays (active low)
```

```c
      activatePump(DISABLE_PUMPS);
      output_low(RELAY_ENABLE);

   // Main program that is executed endlessly
   moneyInserted = true; // Turns on this flag regardless the past events.
   while (true)
   {
      // Waiting for user to select a valid product and insert enough coins
      key = NULL_CHAR;
      enable_interrupts(GLOBAL);
      enable_interrupts(INT_TIMER1);
      // This part cannot be done by chooseProduct()
      // function because this includes more functionality
      while (VALUE_IS_NOT_A_PRODUCT(key))
      {
         showMessage(1, 0, 1); // "Insert money"
         while (true)
         {
            keypad();
         }
         if (VALUE_IS_A_PRODUCT(key))
         {
            selectedProduct = ASCII_TO_INDEX(key);
            if (productAvailable[selectedProduct] == false) // If the product is not available
            {
               showMessage(3, 1, 1); // "Product not available"
               key = NULL_CHAR; // Avoid escaping loop
            }
            // If the credits are insufficient to purchase the selected product
            else if (productPrice[selectedProduct] > credit)
            {
               showMessage(4, 1, 1); // "Insufficient credit"
               key = NULL_CHAR; // Avoid escaping loop
            }
         }
      }
      // Once the user has selected an available product...
      // Decrease the current number of credits by the price of the purchased product
      credit -= productPrice[selectedProduct];
      showNames(productNames[selectedProduct]);// Display the name of the selected product
      showMessage(5, 1, 0); // "selected"
      showMessage(6, 0, 1); // "Dispensing product"
      // Disable all interrupts to avoid any delay
      disable_interrupts(GLOBAL);
      // Activate the corresponding pump
      activatePump(selectedProduct + 1);
      // Individual calibrated time for each product
      for (uint8_t i = 0; i < productTimer[selectedProduct]; i++)
      {
         tinyDelay();
         tinyDelay();
      }
      // And after a certain time, turn off that pump
      activatePump(DISABLE_PUMPS);
      // Display a farewell message and restart the main program
      showMessage(7, 1, 1); // "Thanks for buying"
   }
} // end of main function

////////// Custom functions //////////
int8 scanKeypad(void) // STATUS: ACTIVE
{
   for (int8 row = 0; row < ROWS_NUMBER; row++)
   {
      for (int8 column = 0; column < COLUMNS_NUMBER; column++)
      {
         output_low(COLUMNS[column]);
         if (input(ROWS[row]) == 0)
         {
            output_float(COLUMNS[column]);
            return matrix[row][column];
         }
         output_float(COLUMNS[column]);
      }
   }
   return NULL_CHAR;
}

void keypad(void) // STATUS: ACTIVE
{
   while (true) // waits until user presses a key
   {
      if (key != scanKeypad())
      {
         tinyDelay();
         key = scanKeypad();
         if (key != NULL_CHAR)
         {
            return;
         }
      }
      if (moneyInserted) // If user has inserted money...
      {
         moneyInserted = false; // Clear that flag
         lcd_gotoxy(1, 2); // And display the current credit on the screen
         printf(lcd_putc, "$%u  ", credit);
      }
   }
}

void activatePump(int val) // STATUS: ACTIVE
{
   if (val < 1 || val > SHIFT_REGISTER_OUTPUTS) // Reject if value is unvalid
      return;
   for (int i = SHIFT_REGISTER_OUTPUTS; i > 0; i--) // For every bit...
   {
      if (i == val) // If this bit should be on...
      {
         output_low(SER); // Send 0 (active low)
      }
      else
      {
         output_high(SER); // Otherwise, send 1 (active low)
      }
      output_low(CLK); // Shift the bit
      output_high(CLK);
   }
   output_low(LATCH); // And charge them to the outputs at the end
   output_high(LATCH);
}

void stdDelay(void) // STATUS: ACTIVE
{
   delay_ms(STD_DELAY); // Avoid inlining
}

void tinyDelay(void) // STATUS: ACTIVE
{
   delay_ms(TINY_DELAY); // Avoid inlining
}

void printStringROM(int16 ptr) // STATUS: ACTIVE
{
   uint8_t temp; //
   while (true)
   {
      read_program_memory(ptr++, &temp, 1); // Read program memory byte by byte
      if (temp == 0x00) // If it's a null character (end of string), return
      {
         return;
      }
      lcd_putc(temp); // Otherwise print that character
   }
}

void clearScreen(void) // STATUS: ACTIVE
{
   lcd_putc("\f");
}

void showNames(int16 namePtr) // STATUS: ACTIVE
{
   clearScreen();
   printStringROM(namePtr); // Shows name
}

void showMessage(uint8_t message, int1 delay = 1, int1 clear = 1) // STATUS: ACTIVE
{
   if (clear)
   {
      clearScreen(); // Clear screen before printing the message
   }
   switch (message)
   {
      case 0:
      #ifdef ENGLISH
         lcd_putc("Starting\nmachine");
      #else
         lcd_putc("Iniciando\nmaquina");
      #endif
         break;
      case 1:
      #ifdef ENGLISH
      printf(lcd_putc, "Insert coins:\n$%u", credit);
      #else
      printf(lcd_putc, "Inserte dinero:\n$%u", credit);
      #endif
         break;
      case 3:
      #ifdef ENGLISH
      lcd_putc("Product not\navailable");
      #else
      lcd_putc("Producto no\ndisponible");
      #endif
```

```
            break;
        case 4:
        #ifdef ENGLISH
        lcd_putc("Insufficient\ncredit");
        #else
        lcd_putc("Credito\ninsuficiente");
        #endif
            break;
        case 5:
        #ifdef ENGLISH
        lcd_putc("\nselected");
        #else
        lcd_putc("\nseleccionado");
        #endif
            break;
        case 6:
        #ifdef ENGLISH
        lcd_putc("Dispensing\nproduct");
        #else
        lcd_putc("Llenando\nproducto");
        #endif
            break;
        case 7:
        #ifdef ENGLISH
        lcd_putc("Thanks for\nbuying");
        #else
        lcd_putc("Vuelva\npronto");
        #endif
            break;
        case 20:
        #ifdef ENGLISH
        lcd_putc("\nchosen");
        #else
        lcd_putc("\nelegido");
        #endif
            break;
    }
    if (delay)
    {
        stdDelay(); // Wait
    }
}

```

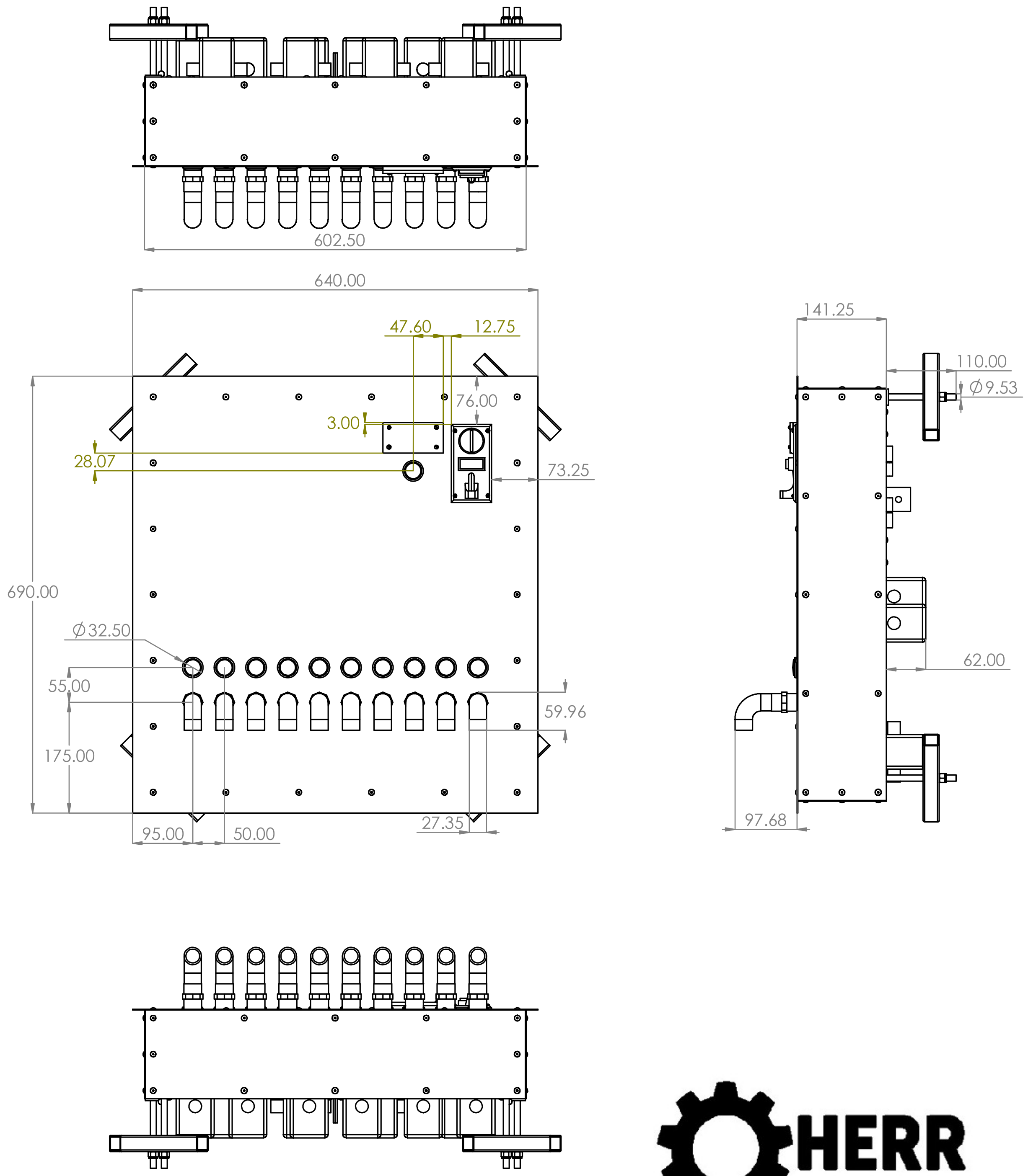


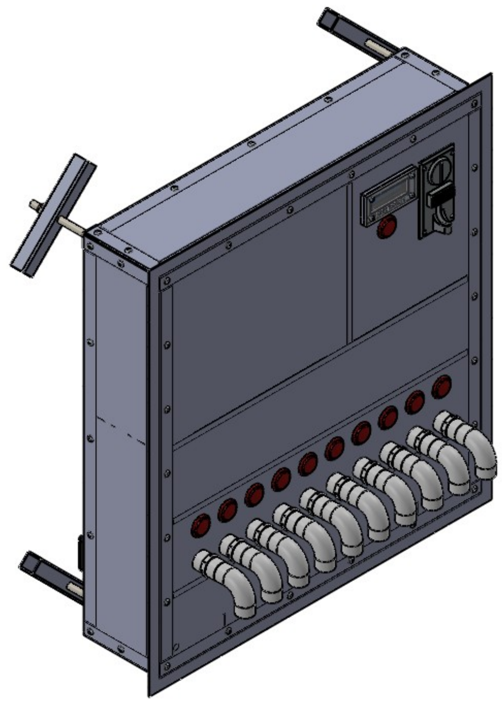
E-mail: contactotransistorizedmx@gmail.com
Designed by Herr Francis

Sheet: /
File: VendingMachineMK1.kicad_sch

Title: Electronic diagram for vending machine v1.0

Size: User | Date: 2026-7-1 | Rev:
KiCad E.D.A. 9.0.6 | Id: 1/1

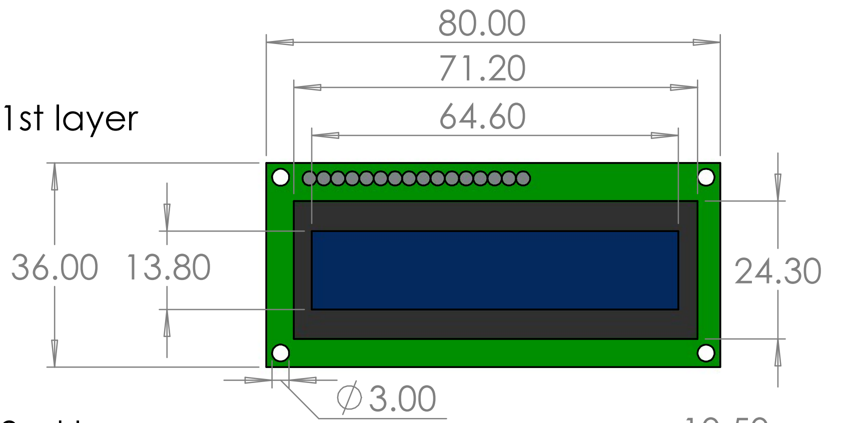


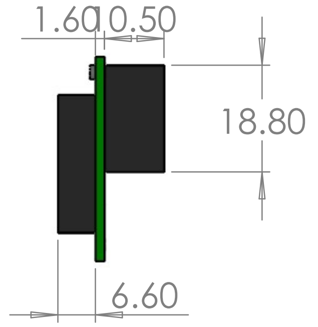


HERR TECHNIK

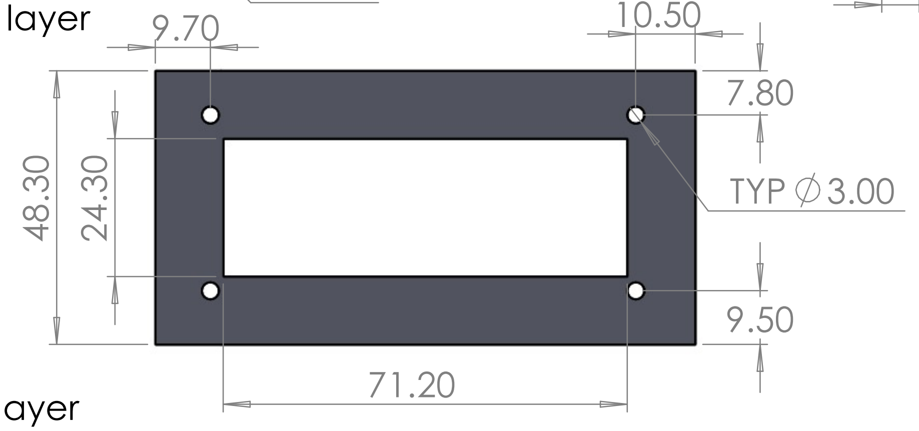
COMPLETE ASSEMBLY	
Designer	Herr Francis
Company	
Date	January 9th, 2026
Price	
Weight	
Class	

1st layer

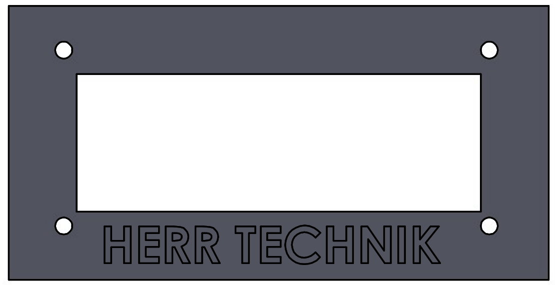




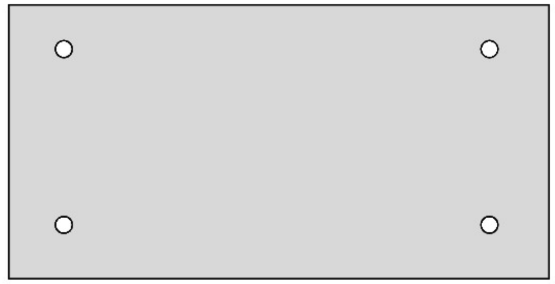
2nd layer

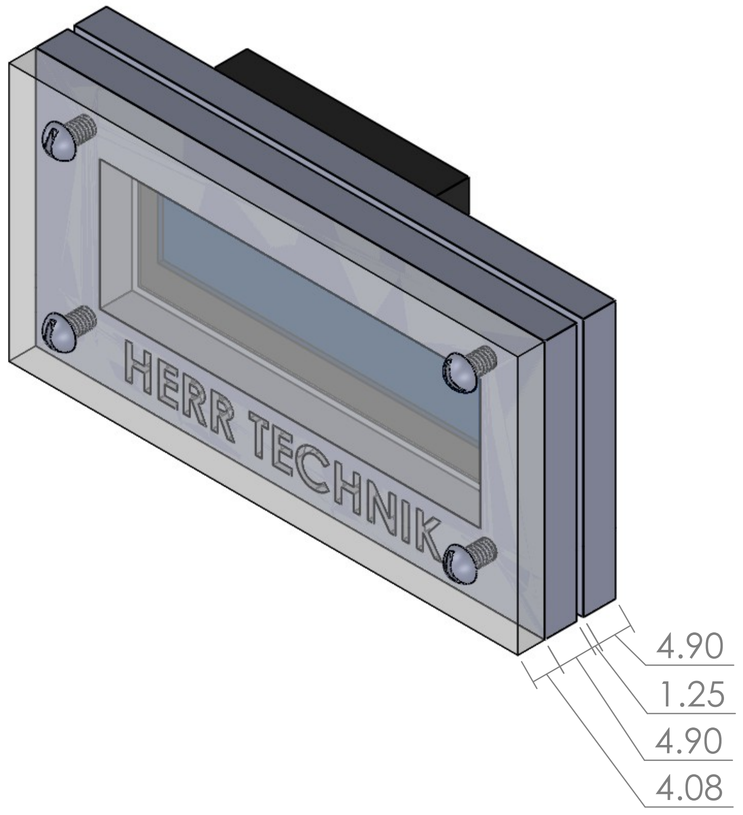


3rd layer



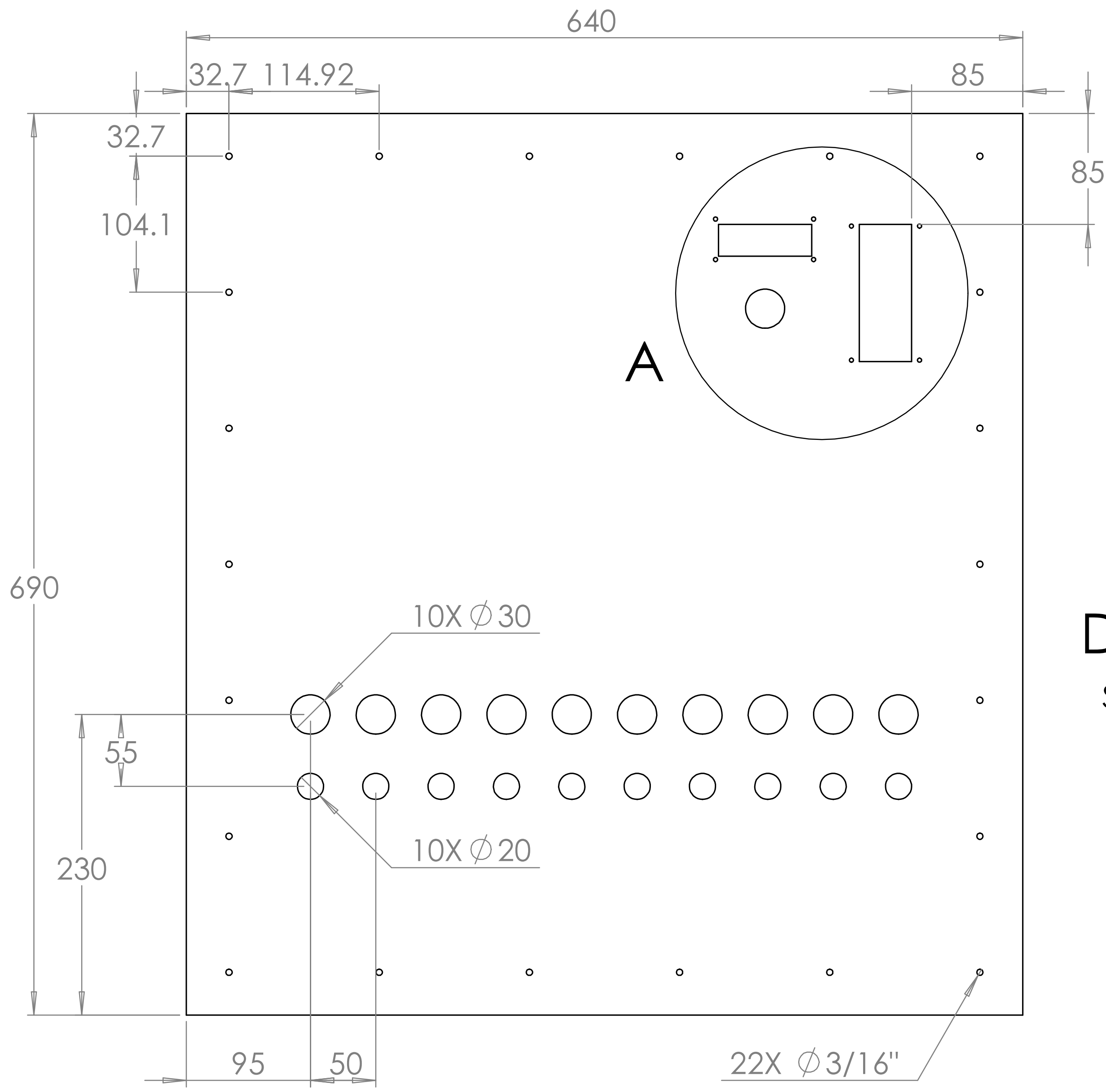
4th layer

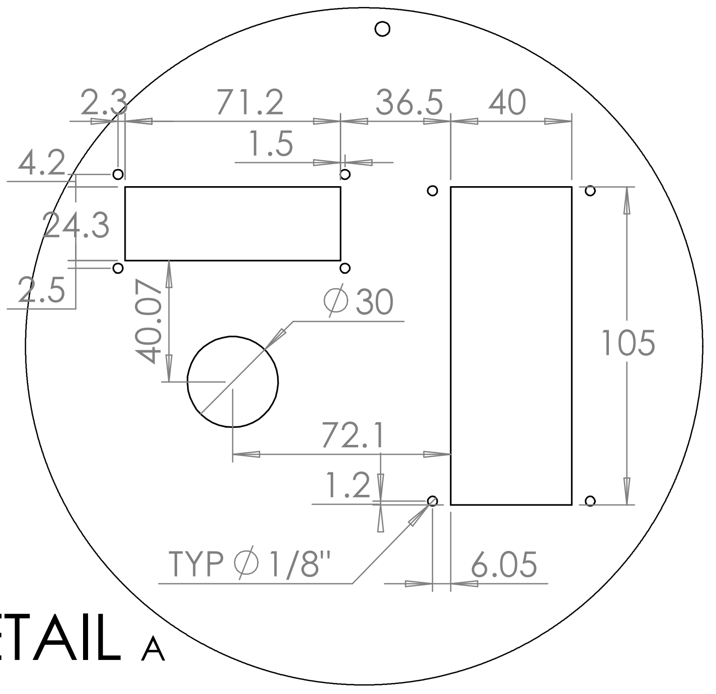




HERR TECHNIK

LCD protective shield	
Designer	Herr Francis
Company	
Date	January 9th, 2026
Price	
Weight	
Class	



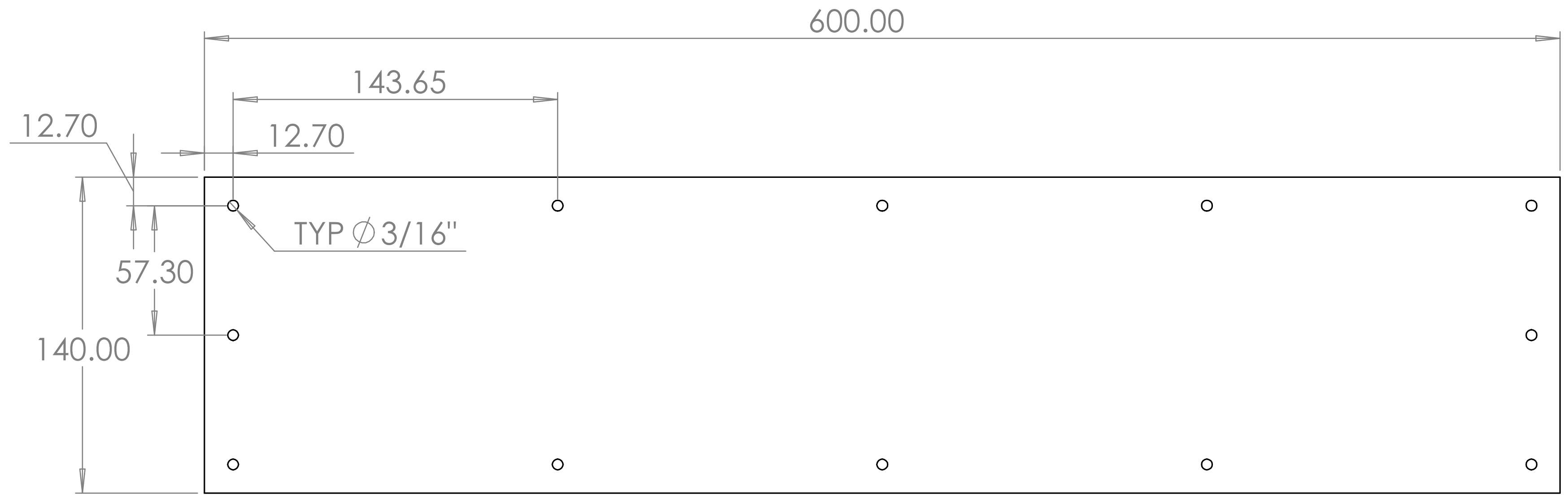


DETAIL A

SCALE 2 : 5

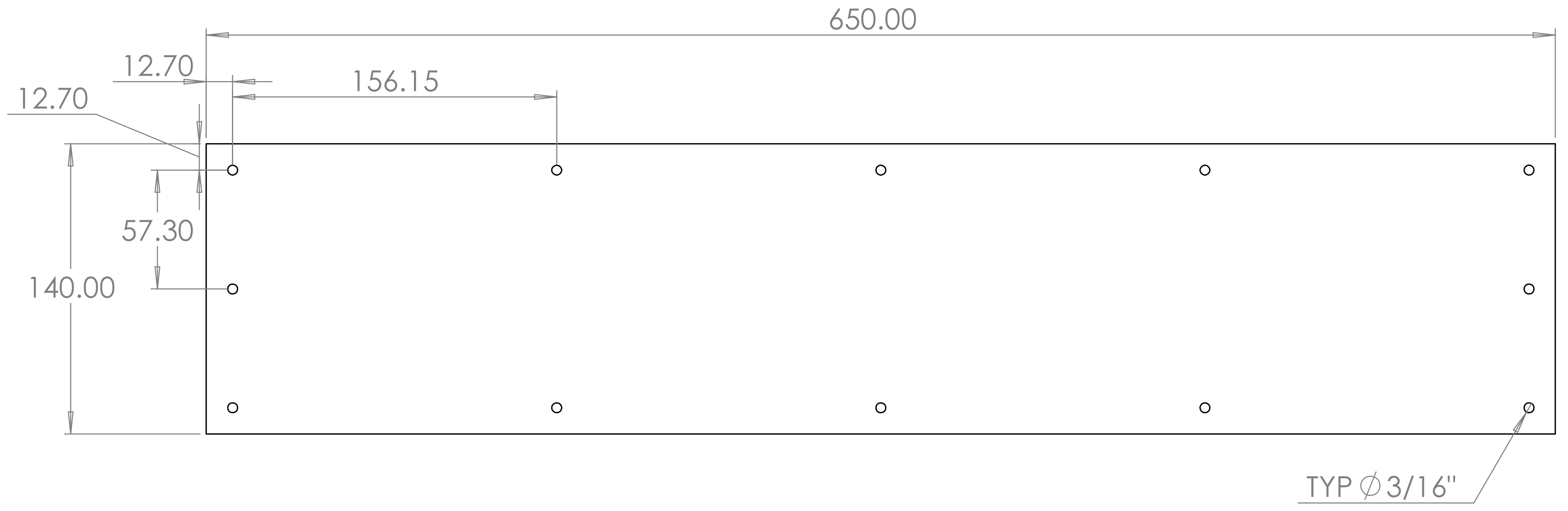
Material: 18-gauge galvanized steel sheet
Thickness: 1.25 mm

Front panel	
Designer	Herr Francis
Company	
Date	January 9th, 2026
Weight	
Class	



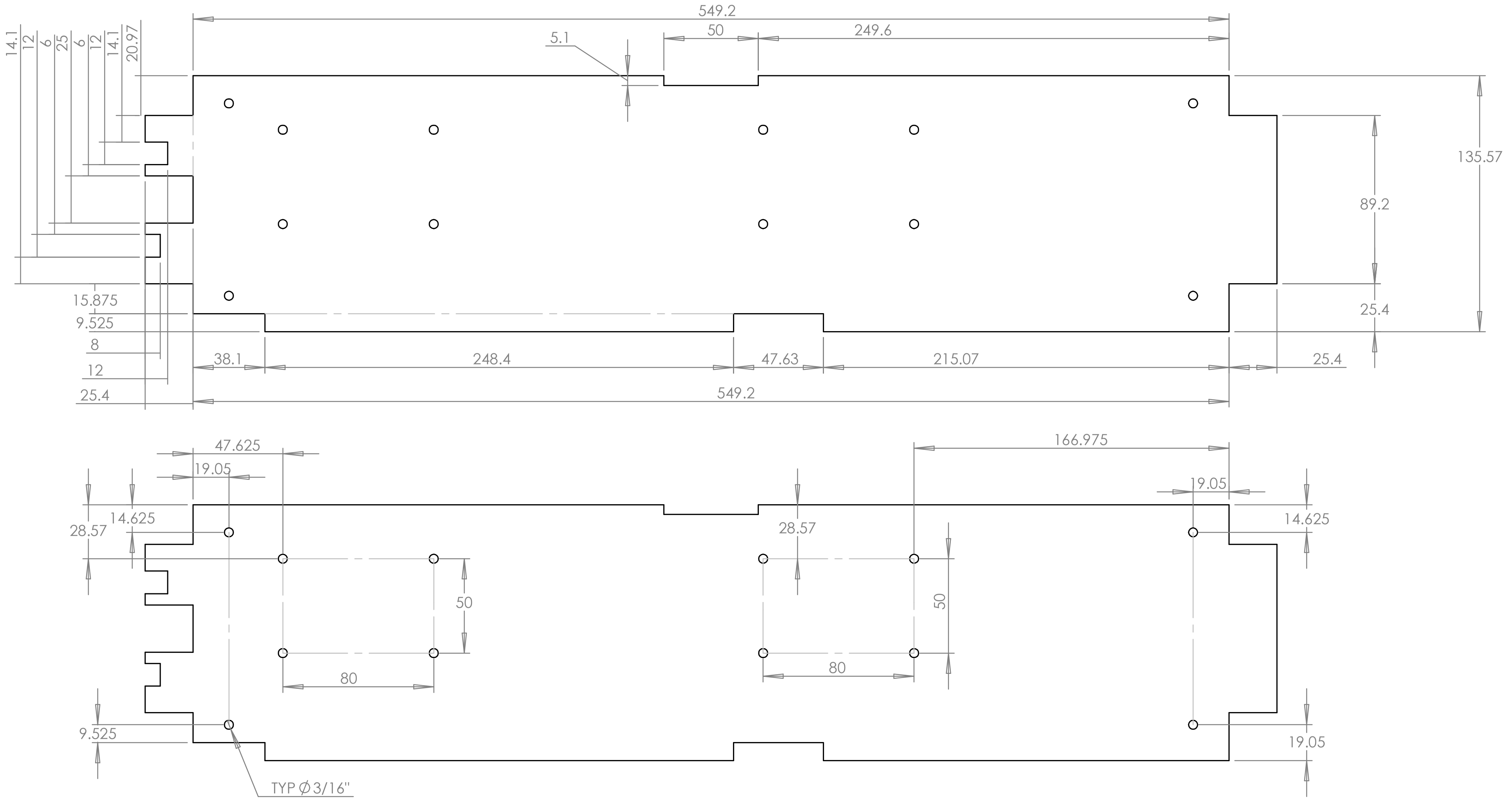
Material: 18-gauge galvanized steel sheet
Thickness: 1.25 mm

TOP AND BOTTOM METAL PANELS	
Designer	Herr Francis
Company	
Date	January 9th, 2026
Price	
Weight	
Class	



Material: 18-gauge galvanized steel sheet
Thickness: 1.25 mm

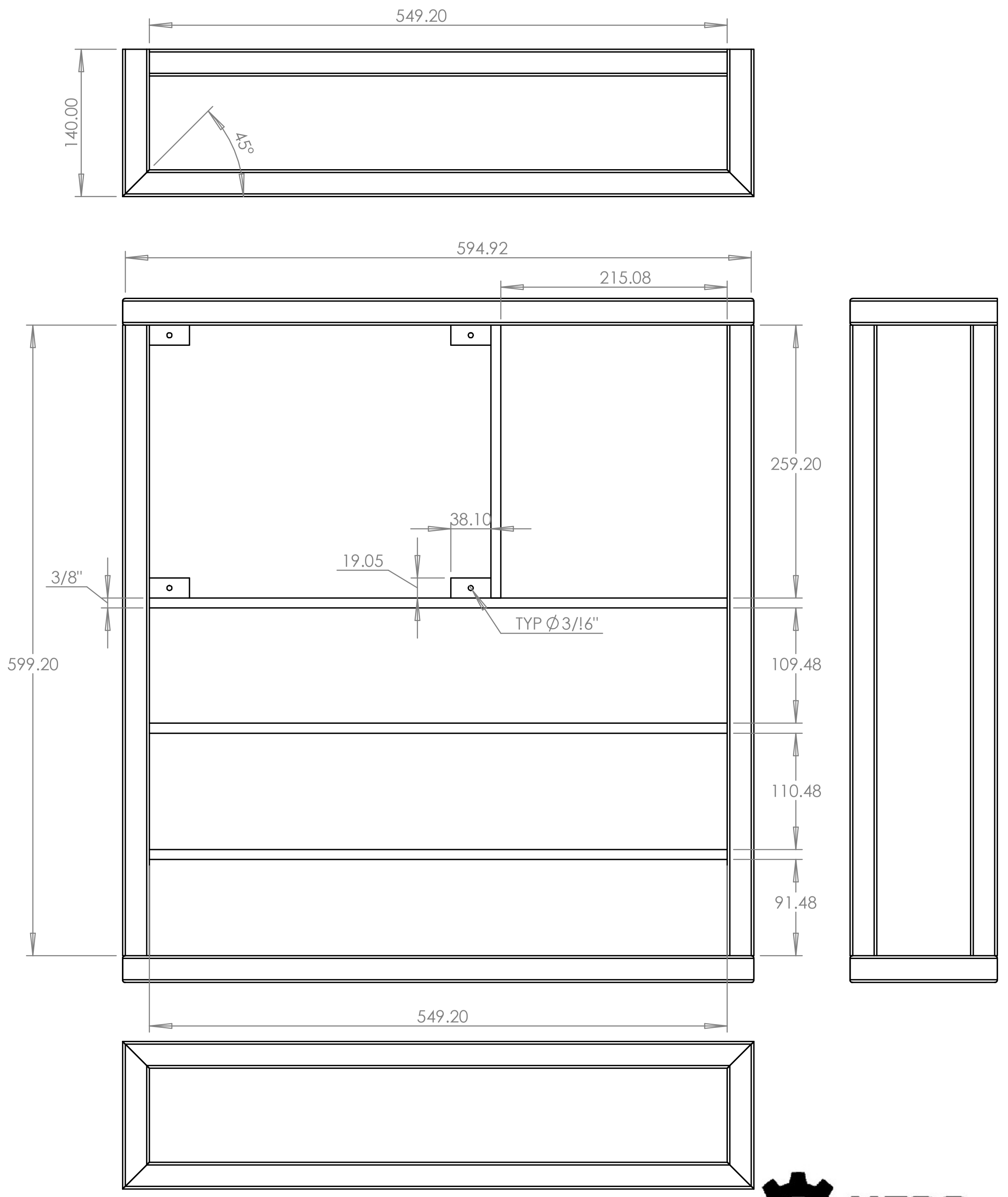
SIDE METAL PANELS	
Designer	Herr Francis
Company	
Date	January 9th, 2026
Price	
Weight	
Class	

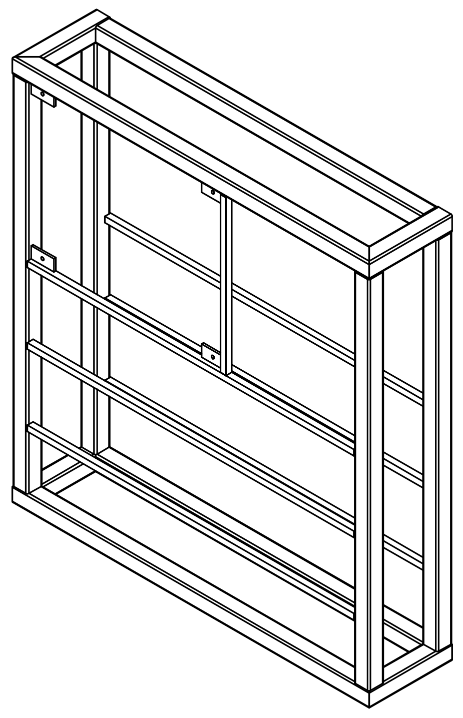


Material: 18-gauge galvanized steel sheet
Thickness: 1.25 mm

HERR TECHNIK

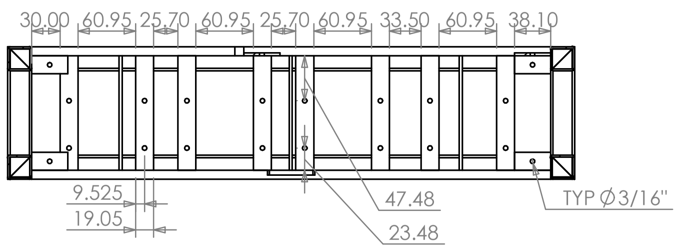
Internal protective metal panel	
Designer	Herr Francis
Company	
Date	January 9th, 2026
Price	
Weight	
Class	

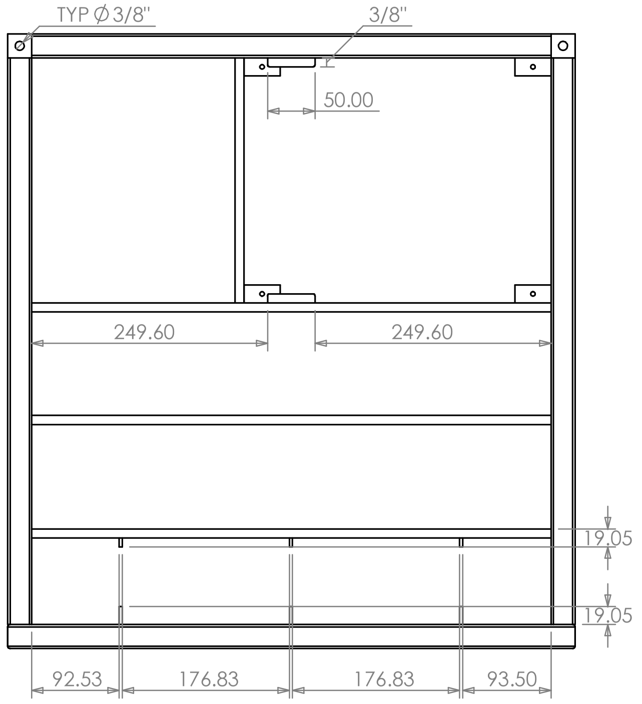




14 gauge 1" X 1" Square piping
3/8 square bar
3/4 flat bar

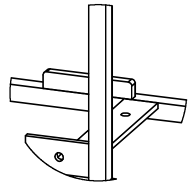
BASIC STRUCTURE	
Designer	Herr Francis
Company	
Date	January 9th, 2026
Price	
Weight	
Class	

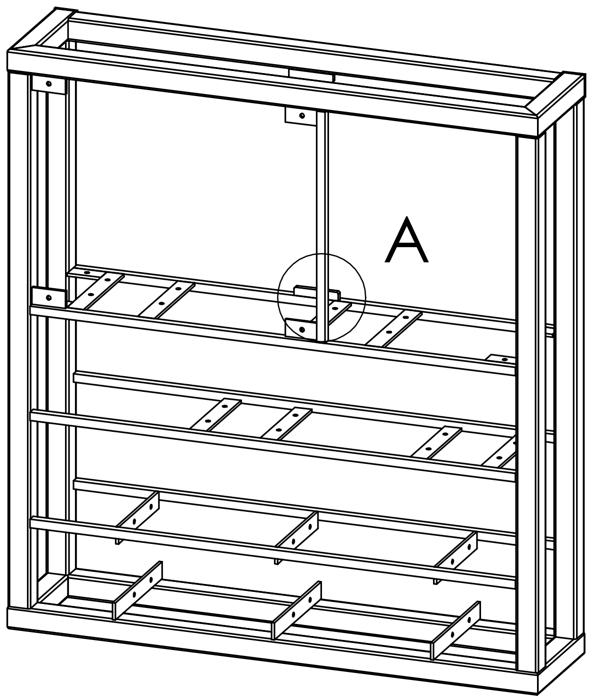




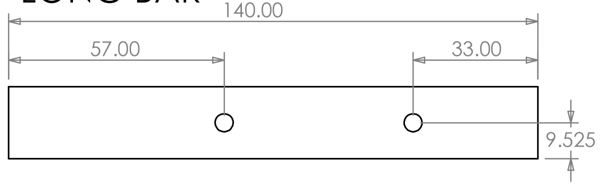
DETAIL A

SCALE 1 : 4

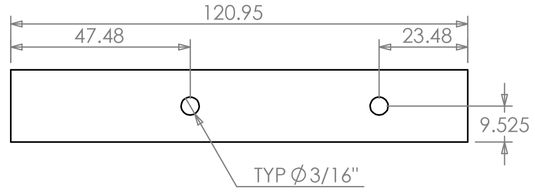




LONG BAR

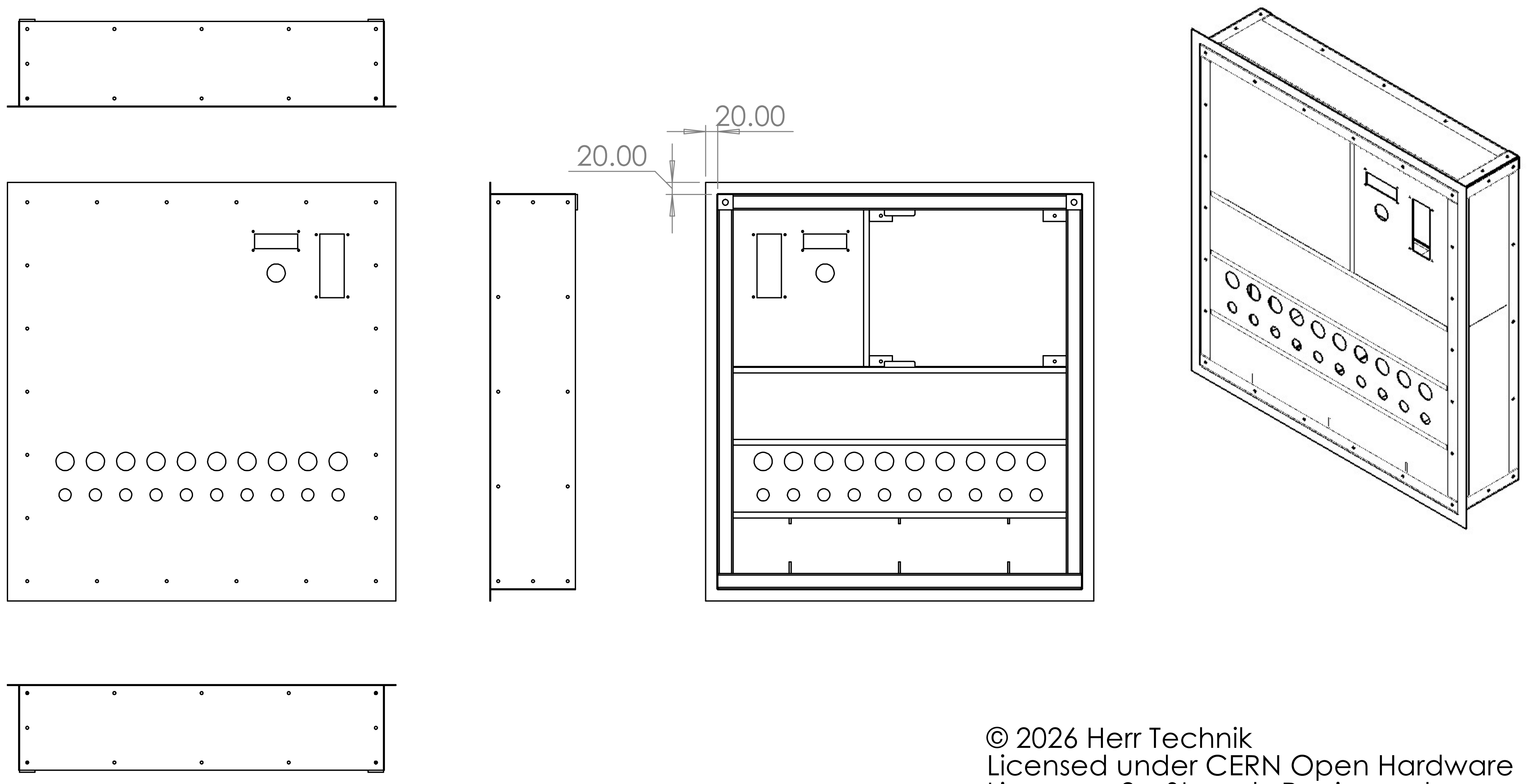


SHORT BAR

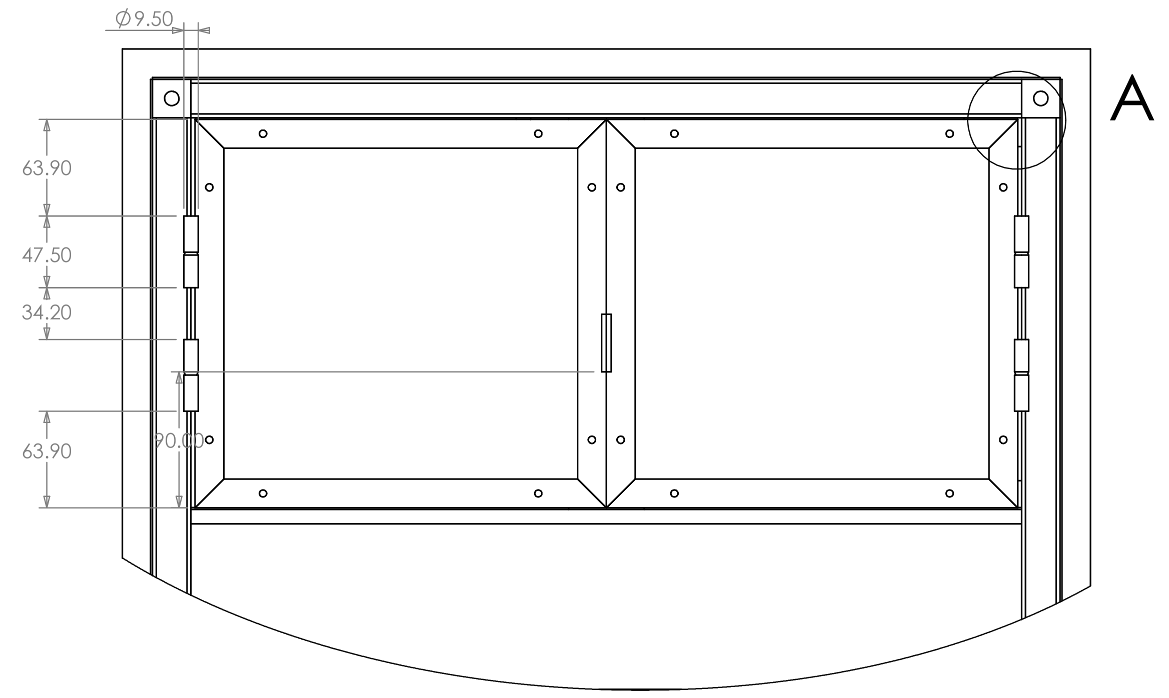


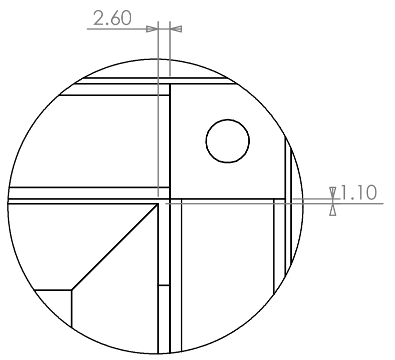
3/4" X 1/8 "FLAT BARS	
Designer	Herr Francis
Company	
Date	January 9th, 2026
Price	
Weight	
Class	

HERR TECHNIK



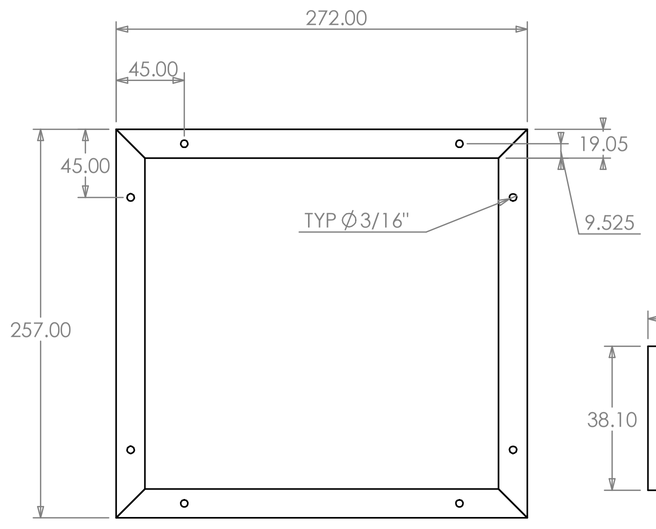
METAL PANELS	
Designer	Herr Francis
Company	
Date	January 9th, 2026
Price	HERR TECHNIK
Weight	
Class	

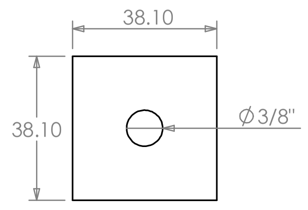


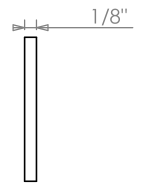


DETAIL A

SCALE 3 : 5







DOORS	
Designer	Herr Francis
Company	
Date	January 9th, 2026
Price	
Weight	
Class	