ITS ICT
Accademia Digitale Liguria

CONTENUTI	• Aggiungi qui i titoli delle sezioni. Sono utili dei brevi riepiloghi! • Consiglio: compila questa sezione dopo aver terminato tutte le altre.

Per iniziare

Questo documento contiene l'intera documentazione tecnica scritta durante le fasi di sviluppo del progetto industriale del corso 12.1, organizzato da ITS - Accademia Digitale Liguria in collaborazione con "Scuola di Robotica".

Introduzione

Questo è un documento di specifica dei requisiti per il software di gestione di Tango, un robot in grado di muoversi, registrare dati dall'ambiente circostante e trasportare carichi. Il controllo del robot da remoto è demandato a una app per smartphone sviluppata appositamente. Il documento in oggetto descrive lo scopo, gli obiettivi e le finalità del nuovo software. Oltre a descrivere i requisiti non funzionali, il medesimo modella i requisiti funzionali, analizza il codice e fornisce anche un manuale d'uso.

Obiettivo del progetto

Obiettivo del progetto industriale è lo sviluppo di un robot completo di varie funzioni, di seguito riportate nelle specifiche tecniche.

Il controllo del robot è demandato a una web app sviluppata appositamente.

Ulteriori risultati attesi sono la messa in gioco efficace delle soft skill proprie del singolo individuo e il miglioramento del team working.

Background

La richiesta di sviluppare questo nuovo software è riferita al progetto industriale che il corso 12.1 di ITS - Accademia Digitale Liguria deve portare a termine secondo curriculum. Dopo attenta analisi, sono stati identificati tre problemi principali per cui trovare una soluzione:

• Controllo del robot: per poter portare a termine i propri compiti, il robot necessita di un'unità di controllo opportunamente programmata

• Registrazione dei dati dall'ambiente circostante: attraverso i sensori montati a bordo macchina, il robot deve essere in grado di registrare dati dall'ambiente circostante e mostrarli su schermo LCD e/o su monitor seriale.

• Movimento del robot: il robot deve essere in grado di muoversi, controllato da remoto, in ogni direzione senza limiti e deve essere altresì in grado di fermarsi in caso di emergenza

Il software è stato sviluppato per rispondere a queste domande e soddisfare le richieste.

Stakeholder

I principali stakeholder per questo progetto includono:

1.

2.

3.

Panoramica del documento

Il resto del documento fornisce le specifiche dettagliate del nuovo sistema. È organizzato come segue:

Requisiti

Overview delle funzioni

- Caratteristiche delle funzioni

Analisi del codice

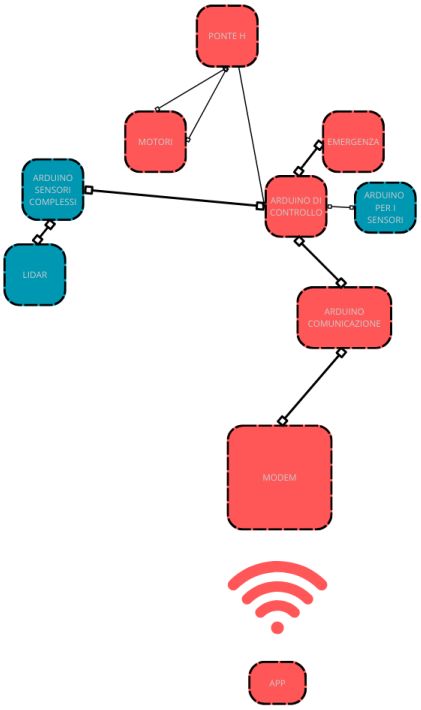
Testing del codice

Appendice - Manuale di utilizzo

Requisiti

Specifiche tecniche

- Robot ATRV-JR
- Arduino Mega 2560 (3 schede)
- App per dispositivi mobili in Java



SCOPO:

Fargli fare un determinato percorso

Gli Arduino devono avere lo stesso standard per poter comunicare ed essere coerenti per poter interfacciarsi con quello di controllo

Tutti i dati lavorano e vengono trasmessi via SERIALE

Il bumper potrebbe avere una doppia gestione

- **Controllo**
- **Emergenza**

LISTA SENSORI:

- **LIDAR**
-

SECONDARIO

PRIORITARIO

Requisiti funzionali

Requisiti non funzionali

Overview delle funzioni

Sono state previste due macro-funzioni, quella di comunicazione e quella di controllo.

Comunicazione

Il sistema di comunicazione è bidirezionale e collega l'applicazione Android e il sistema di controllo basato su Arduino MEGA per controllare il robot.

Applicazione Android

L'interfaccia utente per il controllo di Tango è stata sviluppata utilizzando Android Studio di JetBrains. Questa applicazione Android offre un'interfaccia intuitiva per gli utenti finali, consentendo loro di inviare comandi di controllo all'automobile tramite input touchscreen. Tutti gli input generati dall'applicazione vengono inviati a un server Node.js tramite richieste HTTP POST.

Server Node.js

Il server Node.js funge da intermediario tra l'applicazione Android e Arduino MEGA. È responsabile della ricezione degli input inviati dall'applicazione e dell'inoltro di tali dati al dispositivo Arduino MEGA tramite una connessione Ethernet. Il server è stato implementato per gestire le richieste HTTP POST provenienti dall'applicazione Android e instradare correttamente i dati al dispositivo Arduino MEGA.

Arduino MEGA

Arduino MEGA è il componente centrale del sistema di controllo del robor. È stato equipaggiato con uno shield Ethernet per consentire la comunicazione con il server Node.js e altri dispositivi di rete. Una parte di codice è stata sviluppata per configurare Arduino MEGA come un server utilizzando la libreria Ethernet.h. Questa parte di codice permette la ricezione dei comandi di controllo inviati dal server Node.js per poi eseguirli, controllando così i motori di Tango.

Flusso di comunicazione

Il flusso di comunicazione inizia con l'utente che interagisce con l'applicazione Android tramite input touchscreen. Gli input vengono quindi inviati al server Node.js tramite richieste HTTP POST. Il server Node.js riceve gli input, li elabora e li inoltra al dispositivo Arduino MEGA tramite la connessione Ethernet. Infine, Arduino MEGA interpreta i comandi ricevuti e attua le azioni corrispondenti per controllare i motori del robot.

Controllo

Il sistema di controllo è basato su due schede Arduino MEGA, tre in caso di implementazione di quella dedicata unicamente al LIDAR: la prima scheda gestisce i sensori - ultrasuono per la rilevazione di ostacoli, umidità/temperatura per il rilevamento ambientale - e mostra i dati raccolti sullo schermo LCD ad essa collegata; la seconda gestisce la comunicazione e il controllo del veicolo tramite Arduino MEGA, garantendo la sicurezza e la coerenza del movimento in base ai comandi ricevuti e allo stato dei sensori.

Scheda Arduino - Sensori

Sulla scheda è stato installato un codice che si occupa di monitorare la distanza dagli ostacoli utilizzando sensori ad ultrasuoni, misurare la temperatura e l'umidità ambientale con un sensore DHT11 e monitorare lo stato della batteria tramite un voltmetro.

Le informazioni raccolte vengono quindi trasmesse tramite la comunicazione seriale per essere lette da un dispositivo esterno, come ad esempio un computer, e possono anche essere visualizzate su un display LCD, se il sistema è configurato per farlo.

Di seguito, una breve panoramica delle sue funzionalità.

Gestione della Distanza: Utilizzando sensori ad ultrasuoni, viene misurata la distanza dagli oggetti circostanti e si gestiscono situazioni di emergenza se la distanza è inferiore a una certa soglia.

Gestione della Temperatura e dell'Umidità: Viene utilizzato un sensore DHT11 per misurare la temperatura e l'umidità ambiente e questi fornisce le informazioni tramite la comunicazione seriale.

Monitoraggio dello Stato della Batteria: Lo stato della batteria viene monitorato misurando la tensione da un voltmetro e viene trasmesso tramite la comunicazione seriale.

Loop Principale: Nel loop principale del programma, vengono gestite le operazioni di monitoraggio della distanza, aggiornamento del display LCD e monitoraggio periodico della temperatura, umidità e tensione della batteria.

Gestione del Tempo: Vengono eseguite determinate azioni ogni tot minuti utilizzando il modulo millis() per calcolare il tempo trascorso.

Scheda Arduino - Controllo e Comunicazione

Si tratta del firmware per il controllo di Tango tramite un Arduino MEGA, che permette la gestione della comunicazione con altri dispositivi e sensori. Di seguito, una breve panoramica delle sue funzionalità:

Configurazione e Inizializzazione: In questa sezione, vengono dichiarate le variabili, configurati i pin di input/output e inizializzate le comunicazioni seriali con altri dispositivi.

Loop Principale: Il programma esegue continuamente un ciclo in cui controlla la comunicazione seriale, mappa i messaggi ricevuti, gestisce lo stato di emergenza e il movimento del veicolo.

Gestione dell'Emergenza: Viene definita una funzione per gestire lo stato di emergenza del veicolo, che ferma il veicolo in caso di emergenza e invia segnali agli altri dispositivi.

Comunicazione Seriale: Una funzione per gestire la comunicazione seriale, sia in entrata che in uscita, con altri dispositivi e sensori.

Gestione del Movimento: La funzione movement() gestisce il movimento del veicolo in base ai comandi ricevuti. Utilizza un'interruzione per rilevare lo stato di emergenza e reagisce di conseguenza.

Controllo della Velocità: Ci sono funzioni per controllare la velocità del veicolo e garantire che rimanga entro limiti sicuri.

Gestione della Rotazione: Viene controllata la rotazione del veicolo e vengono prese misure per evitare movimenti indesiderati o pericolosi.

Funzioni Ausiliarie: Alcune funzioni ausiliarie gestiscono operazioni specifiche, come l'arresto dei motori o la decelerazione graduale.

Sviluppo - CODICE

L’analisi del codice permette una migliore comprensione dei passi fatti nello sviluppo e delle buone pratiche impiegate.

ArduinoServer.ino:

Questo framework è stato progettato in modo da gestire la connettività di rete Ethernet, il controllo di emergenza e la comunicazione con un altro dispositivo Arduino tramite protocollo HTTP.

Dipendenze

Il codice utilizza le seguenti librerie:

- **SPI**: Comunicazione SPI con la scheda Ethernet.
- **Ethernet**: Gestione della connessione Ethernet.
- **String**: Manipolazione delle stringhe.
- **ArduinoJson**: Analisi dei dati JSON ricevuti tramite le richieste HTTP.

```
#include <SPI.h>
#include <Ethernet.h>
#include <string.h>
#include <ArduinoJson.h>
```

Configurazione Iniziale

Di seguito, vengono definiti i pin utilizzati e le variabili per la gestione della connettività e dello stato di comunicazione. Vengono quindi impostate le informazioni di rete, tra cui l'indirizzo MAC, l'indirizzo IP del dispositivo Arduino e l'indirizzo IP del server remoto.

```
//PINS
int emergency = 13;

// Connection checks variables
unsigned long startTime = millis();
unsigned long currentTime;
unsigned long duration = 4500;
bool checked = false;
bool failed = false;
bool comExtableshed = false;

//Connectivity Test IP (Node Server)
int currIPNode = 2;

//Arduino Server
int arduinoIP = 4;
int arduinoPort = 80;

// Network Settings
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };  // MAC address
IPAddress ip(192, 168, 1, arduinoIP); // IP address
IPAddress nodeServer(192, 168, 1, currIPNode);  // IP address of the node server

EthernetClient client;
EthernetServer server(80);
```

Network Settings:

Per quanto riguarda i parametri di configurazione di rete per la scheda Arduino Mega, è stata ipotizzata una connessione Ethernet per la comunicazione.

```
// Network Settings

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };  // MAC address

IPAddress ip(192, 168, 1, arduinoIP); // IP address

IPAddress nodeServer(192, 168, 1, currIPNode);  // IP address of the node server
```

Viene definito l'indirizzo MAC del dispositivo Arduino, univoco e perciò non customizzabile.

Per impostare l'indirizzo IP del dispositivo, viene utilizzato l'oggetto IPAddress e i parametri 192, 168, 1 e arduinoIP. Quest'ultimo deve essere sostituito dall'ultimo ottetto dell'indirizzo IP del dispositivo Arduino in rete.

Per definire l'indirizzo IP del server nodo, con cui l'Arduino comunicherà, viene utilizzato anche in questo caso l'oggetto IPAddress, con 192, 168, 1 come parte fissa dell'indirizzo IP e currIPNode come variabile che rappresenta l'ultimo ottetto dell'indirizzo IP del server nodo all'interno della rete.

Setup()

Nella funzione di setup, vengono inizializzati i pin, la comunicazione seriale e la connessione Ethernet.

```
void setup() {

  // emergency PIN

  pinMode(13, OUTPUT);

  unsigned long startTime = millis();

  Serial.begin(9600);

  // -- Start the Ethernet connection --

  Ethernet.begin(mac);

  Ethernet.setLocalIP(ip);
  // -- Testing connectivity --  Serial.println(">> Testing connectivity (using Node server connection on port 3000)");

    if (Ethernet.linkStatus() == LinkOFF){

    Serial.println("Ethernet cable is not connected!");

  }
```

Viene quindi eseguito un test di connettività con il server remoto usando i retry e viene avviato il server Arduino.

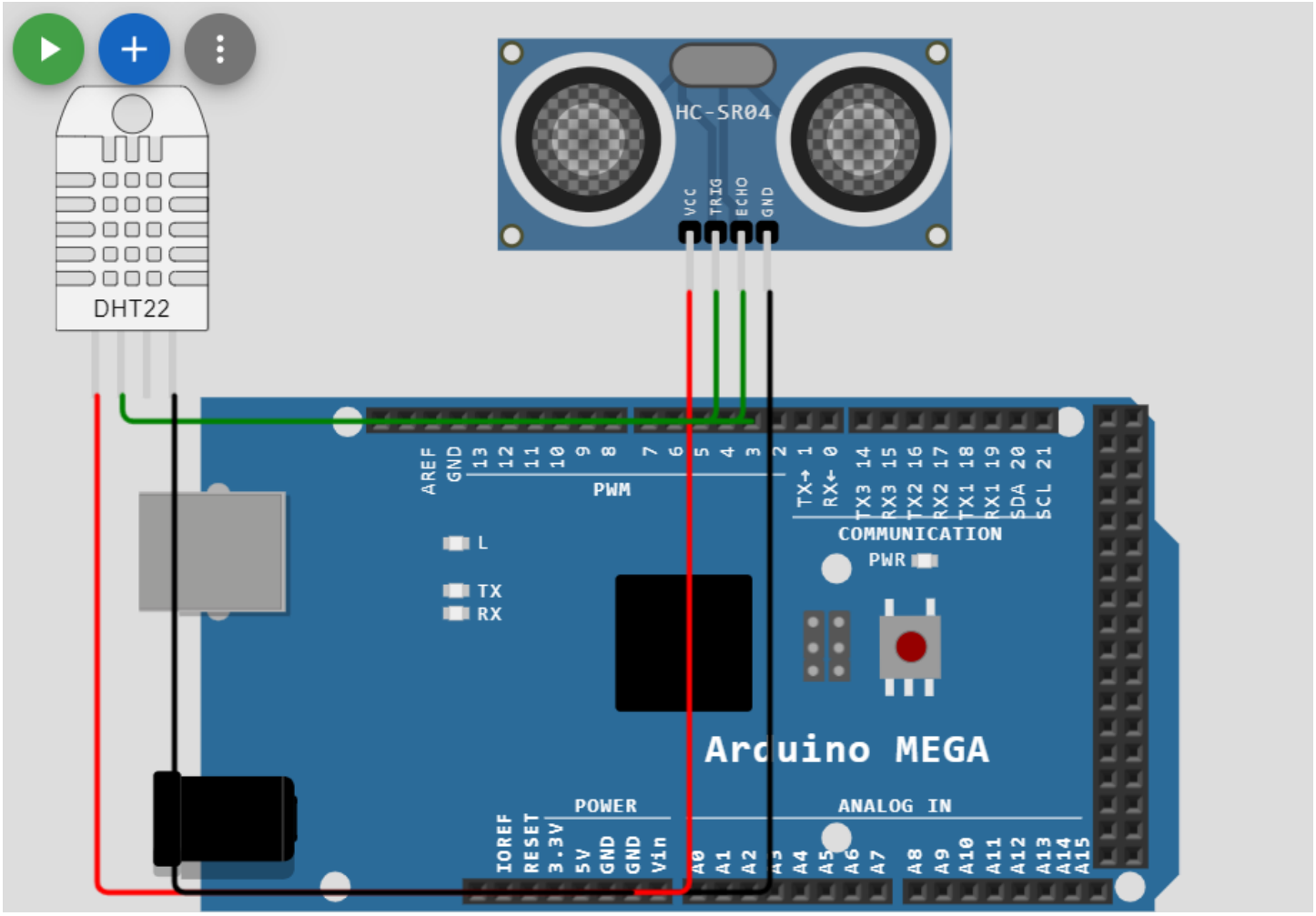
```
//
 // -- HTTP GET request to test connectivity (using retries) --
 //
 for (int retryCount = 0; retryCount < 3; retryCount++){
  if (client.connect(nodeServer, 3000)) {
    /*DEBUG: */Serial.println("Connected to server");

    client.println("GET /");
    String connectionString = String("Host: 192.168.0." + String(currIPNode));
    client.println(connectionString);   // Replace with the actual IP address or domain of Node server
    client.println("Connection: keep-alive");
    client.println();
    break;
  } else{
    if (retryCount == 2){
      /*DEBUG: */Serial.println("Connection failed!");
    } else{
      /*DEBUG: */Serial.println("Connection failed. Retrying...");
      delay(3000);
    }
  }
 }
 client.stop();
```

Funzioni Ausiliarie

- **startEmergencyStop()**: Invia un segnale di emergenza per interrompere il funzionamento del sistema.
- **forwardToControlArduino()**: Invia una stringa di direzione all'altro Arduino.emerge

Loop()

La funzione loop gestisce continuamente la connessione con il client, riceve e analizza i dati inviati tramite richieste HTTP POST e gestisce il controllo di emergenza. Viene anche letta la comunicazione seriale dall'altro Arduino per acquisire i valori dei sensori.



App:

fragment_home.hml

-ConstraintLayout: È il layout principale e contiene tutti gli altri elementi. Utilizza il sistema di vincoli (app:layout_constraint*) per posizionare gli elementi in relazione l'uno all'altro.

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/rootLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".HomeFragment"
    android:background="@color/lightgray">
```

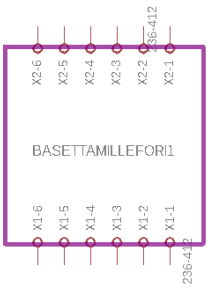
-FrameLayout (topFrame): È un contenitore che funge da sfondo principale. Contiene un TextView centrato con il testo "Your Text Here". Ha un'immagine di sfondo (@drawable/frame_border) e una altezza di 310dp.

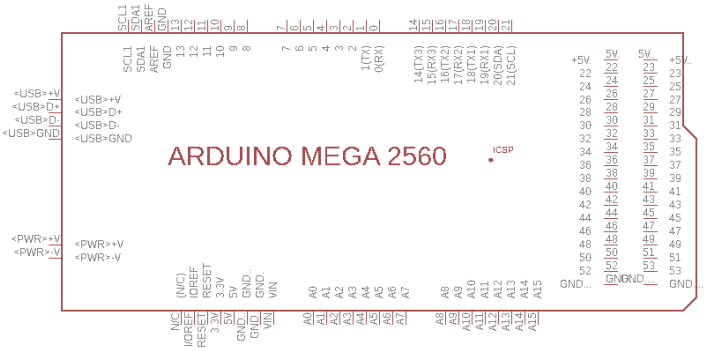
Sviluppo - STAMPA E GRAFICA

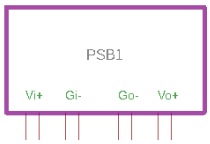
1. Creazione di render per la stampa 3D e con lasercut di componenti
2. Uso di programmi come Xtool Creative Space (XCS) e Sharp3D

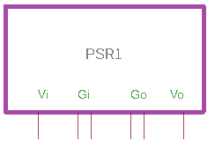
Sviluppo - HARDWARE

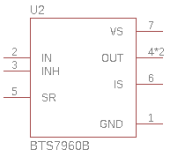
1. Montaggio dei componenti hardware a bordo macchina
2. Schema elettrico

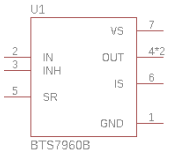

X2-6
X2-5
X2-4
X2-3
X2-2
X2-1
236-412
BASETTAMILLEFORI1
X1-6
X1-5
X1-4
X1-3
X1-2
X1-1
236-412


ARDUINO MEGA 2560
ICSP
<USB>+V
<USB>D+
<USB>D-
<USB>GND
<PWR>+V
<PWR>-V


PSB1
Vi+
Gi-
Go-
Vo+


PSR1
Vi
Gi
Go
Vo


U2
IN
INH
SR
VS
OUT
IS
GND
4*2
BTS7960B


U1
IN
INH
SR
VS
OUT
IS
GND
4*2
BTS7960B


CHIAVE1