Progetto MacchinaLego

Documento Analisi e Progettazione

Cliente: Prof.ssa Sara Valenti

Fornitore: KAR S.P.A

Data: 8/04/2024

* Scopo del Progetto:

Si intende realizzare una macchina lego telecomandata in due **modalità**: la prima via **WI-FI** con un software/applicazione programmata dai sottoscritti e la seconda via **AUTOMATICA/AUTONOMA**.

La macchinina è una **costruzione di Lego unita con Arduino** la quale sarà dotata di funzionalità del movimento e di tutte le altre caratteristiche sotto elencate.

* Specifica dei Requisiti:

Lato Software:

-Buzzer: Utilizzeremo un **buzzer** per la simulazione del clacson della macchina quando il tasto “**C**” della tastiera sarà premuto.

Lato Hardware:

-Buzzer: Utilizzeremo un **buzzer** che collegheremo all’Arduino.

Lato Software:

-Calcolo della Velocità: Verrà visualizzata la **velocità attuale** sullo schermo.

Lato Hardware:

-Calcolo della Velocità: Verrà visualizzata la **velocità attuale** della macchina sulla base delle informazioni dateci dalla libreria **EVShield.** (e anche di nostra scelta)

Lato Software:

-Angolo della Sterzata: Verrà visualizzato **l’angolo della sterzata** attuale delle ruote.

Lato Hardware:

-Angolo della Sterzata: Verrà calcolato **l’angolo della sterzata** in base a quanti gradi vengono girate le ruote anteriori.

Lato Software:

-Distanza del pericolo: Verrà visualizzata la **distanza dal sensore ad ultra-suoni** fino all’ostacolo visto dalla macchina.

Lato Hardware:

-Sensore ad Ultra-Suoni: Sensore (**anteriore**) che permetterà alla **macchina di evitare di andare a sbattere contro ostacoli**, facendosi che la macchina abbia un intelligenza propria.

Lato Software:

-Distanza del pericolo: Verrà visualizzata la **distanza dal sensore di movimento** fino all’ostacolo visto dalla macchina.

Lato Hardware:

-Sensore di Distanza: Sensore (**posteriore**) che permetterà alla macchina di evitare di andare contro gli ostacoli in due modalità:

Nella **prima modalità** cioè quella **telecomandata,** tramite azione dell’utente, in direzione **posteriore**.

Nella **seconda modalità** cioè quella **automatica**, la macchina eseguirà un percorso deciso da essa (come se avesse un **intelligenza propria**), la quale riuscirà ad evitare ogni tipo di ostacolo come nella prima modalità, in direzione **posteriore**.

* Progettazione

Elenco classi da realizzare

Immagine che contiene schermata, testo, Software multimediale, Software per la grafica

Descrizione generata automaticamente

Descrizioni delle classi:

**Package automaticScreen:**

**AutoScreen:**

Questa classe gestisce la schermata di guida autonoma del progetto.

Attributi:

-**private PApplet a:**questo attributo permette l’utilizzo dei metodi della libreria Processing.core.\*

- **private Object s:** questo attributo permette l’utilizzo della seriale

- **private boolean entry:** questo attributo ci permette di capire quando vengono rispettate determinate condizioni.

- **private float previousMillis:** viene utilizzata per tenere traccia del tempo trascorso dall'ultimo evento significativo.

- **private int dir:** viene utilizzato per memorizzare la direzione angolare della macchinina, ovvero la direzione in cui si sta muovendo.

- **private int x, y:** Questi attributi memorizzano le coordinate della posizione della

Macchinina.

- **private int pDir**: Questo attributo memorizza la direzione precedente della macchinina.

- **private final int START\_X,START\_Y:** questi attributi servono per far capire in quale parte dello schermo far partire il disegno dello spostamento della macchina.

- **private List<Integer[]> pos**: questo attributo viene utilizzato per memorizzare le posizioni precedenti della macchinina nel corso del suo movimento.

Metodi:

- **public AutoScreen(PApplet a,Object s):** questo metodo è il costruttore.

- **public void loop():** Metodo che muovere da sola la macchinina.

- **public void drawInterface():** Metodo che serve a gestire l'aspetto visivo della schermata di guida autonoma.

- **public void drawPath():** Metodo che ha il compito di disegnare il percorso seguito dalla macchinina durante la guida autonoma sulla schermata.

-**public Directions autoPilot():** Metodo che gestisce il controllo del movimento automatico della macchinina.

- **public void clearPos():** Metodo che è responsabile di eliminare tutte le posizioni memorizzate nel percorso della macchinina.

**Directions (Enumerativa):**Enumerativa utile al direzionamento della macchina in un sistema autonomo.

Attributi:

-**private chat cmd:** è un carattere che rappresenta il comando associato a ciascuna direzione della macchina nel sistema autonomo.

-**private int dir:** rappresenta l'angolazione associata a ciascuna direzione della macchina nel sistema autonomo.

Metodi:

-**private Directions(char cmd, int dir):** questo metodo è il costruttore.

-**public chat getCmd():** fa ottenere il carattere direzionale dell’etichetta

-**public int getDir():** fa ottenere l’angolo direzionale dell’etichetta

-**public void setDir(int dir):** imposta la nuova direzione angolare dell’etichetta

**Package bluetoothScreen:**

**RemoteScreen:**

Classe che gestisce la prima schermata del ProgettoLego in modalità remota.Attributi:

-**private PApplet a:**questo attributo permette l’utilizzo dei metodi della libreria Processing.core.\*

-**private Object s:** questo attributo permette l’utilizzo della seriale

-**private KeyStrokesController [] keyStrokes**: contiene i controlli associati ai tasti della tastiera e ai relativi comportamenti che devono essere attivati quando i tasti vengono premuti.

- **private final int BLACK:** serve per la colorazione dei tasti quando non sono stati premuti.

-**private final int BLUE:** serve per la colorazion dei tasti quando sono stati premuti.

-**public static int rounds:** viene utilizzata per memorizzare il numero di "round" associati all'azione corrente del controllo remoto.

-**private int temp:** è utilizzato per memorizzare temporaneamente un valore relativo al numero di round associati all'azione corrente nella schermata di controllo remoto.

Metodi:

-**public RemoteScreen(PApplet a, Object s):** questo metodo è il costruttore

-**public void loop():** metodo che gestisce il ciclo principale di visualizzazione e aggiornamento degli elementi sulla schermata di controllo remoto.

-**public void keyStrokesMod():** metodo chegestisce la visualizzazione e l'interazione con i controlli associati ai tasti della tastiera sulla schermata di controllo remoto.

-**public void keyStrokesReset():**metodo cheviene utilizzato per ripristinare il colore dei controlli della tastiera nella schermata di controllo remoto.

**Package KeyStrokes:**

**KeyStrokesModel:**

Classe MVC che permette la modellizzazione della KeyStrokes.

Attributi:

-**private PApplet a:**questo attributo permette l’utilizzo dei metodi della libreria Processing.core.\*

-**private float x,y:** rappresentano le coordinate (posizione) del tasto sulla schermata.

-**private int color**: rappresenta il colore del tasto sulla schermata di controllo remoto.

-**private int length:** rappresenta la lunghezza del tasto sulla schermata di controllo remoto.

-**private int altitude:** rappresenta l'altezza del tasto sulla schermata di controllo remoto.

-**private String letter:** rappresenta la lettera associata al tasto sulla schermata di controllo remoto.

Metodi:

-**public KeyStrokesModel(PApplet a, float x, float y, int color, int length, int altitude, String letter):** questo metodo è il costruttore

-**public int getAltitude():** metodo che permette di ottenere l’altezza

-**public void setAltitude(int altitude):** metodo che permette di modificare l’altezza  
-**public float getX():** metodo che permette di ottenere la x

-**public void setX(float x):** metodo che permette di modificare la x

-**public float getY():** metodo che permette di ottenere la y

-**public void setY(float y):** metodo che permette di modificare la y

-**public int getColor():** metodo che permette di ottenereil colore

-**public void setColor(int color):** metodo che permette di modificare il colore

- **public int getLength():** metodo che permette di ottenere la lunghezza

-**public void setLength(int length):**metodo che permette di modificare la lunghezza

-**public String getLetter():** metodo che permette di ottenere la lettera del tasto

**KeyStrokesView:**

Classe MVC che permette la visualizzazione della KeyStrokes.

Attributi:

-**private PApplet a:**questo attributo permette l’utilizzo dei metodi della libreria Processing.core.\*

Metodi:

-**public KeyStrokesView(PApplet a):** questo metodo è il costruttore

-**public void key(KeyStrokesModel model):**Metodo che permette la visualizzazione del tasto a schermo

**KeyStrokesController:**

Classe MVC che controlla la KeyStrokes.

Attributi:

-**public KeyStrokesView view:** Rappresenta l'oggetto della vista associato al controllore della tastiera virtuale

-**public KeyStrokesModel model:** rappresenta il modello associato al controllore della tastiera virtuale

Metodi:

-**public KeyStrokesController(PApplet a, float x, float y, int color, int size, int altitude, String letter):**questo metodo è il costruttore

-**public void show():** questo metodo mostra il tasto della keyStrokes a schermo

**Package settingsScreen:**

**Brightness:**

Classe che gestisce l'impostazione della luminosità dello schermo

Attributi:

-**private float standard:** serve per memorizzare il valore predefinito della luminosità dello schermo.  
-**private int bright:** rappresenta il valore corrente della luminosità dello schermo.

-**private PApplet a:**questo attributo permette l’utilizzo dei metodi della libreria Processing.core.\*  
Metodi:

-**public Brightness(PApplet a, int s):**questo metodo è il costruttore

-**public void showBrightness():**metodo che applica la luminosità allo schermo

-**public void more():**metodo che incrementa la luminosità

-**public void less():**metodo che diminuisce la luminosità

-**public void reset():**metodo che resetta la luminosità

-**public void standard(float s):** metodo che imposta la luminosità predefinita

**Settings(Interfaccia):**

Interfaccia per creare una classe che gestisce un impostazione predefinita

Metodi:

-**public void more():**Metodo che incrementa il valore dell'impostazione

-**public void less():**Metodo che decrementa il valore dell'impostazione

-**public void reset():**Metodo che riporta l'impostazione al valore prede

**-public void standard(float s):**Metodo che imposta il valore predefinito dell'impostazione

**SettingsScreen:**

Classe che gestisce la schermata delle Impostazioni nel progetto

Attributi:

-**private PApplet a:**questo attributo permette l’utilizzo dei metodi della libreria Processing.core.\*

-**private Brightness bright:** questo attributo rappresenta il controllo della luminosità dello schermo

-**private Volume vol:** questo attributo rappresenta il controllo del volume dell'applicazione

-**private ControlP5 cp5:**questo attributo rappresenta un oggetto utilizzato per la gestione dei bottoni e degli elementi di interfaccia utente nella libreria ControlP5.

-**private Button[] button:** questo attributo è array di oggetti di tipo Button, che rappresentano i bottoni utilizzati per l'interazione dell'utente nella schermata delle impostazioni.

-**private Vector<PImage>imgs:** questo attributo è un oggetto di tipo Vector che contiene le immagini utilizzate per visualizzare i bottoni nella schermata delle impostazioni.

-**private finali int SIZE:** questo attributo è una costante intera che rappresenta la dimensione dei bottoni nella schermata delle impostazioni.

Metodi:

-**public SettingsScreen(PApplet a, AudioPlayer out, ControlP5 cp5):** qeusto metodo è il costruttore

-**public void loopBrightness():** Metodo che mostra la luminosità

-**public void loop():**Metodo che gestisce la schermata delle impostazioni

-**public void changeBrightness(byte brightness):**Metodo che cambia la luminosità dello schermo

-**public void changeVolume(byte volume):**Metodo che cambia il volume dell'applicazione

-**public void hideAllButton() :** metodo che nasconde tutti i bottoni

-**public void showAllButton(): metodo che mostra tutti i bottoni**

-**public Button getButton(int index):**metodo che restituisce il bottone per implementare la CallBack nella classe che estende PApplet

-**public void isMuoseOver(Button b):** metodo che colora lo sfondo dei bottoni se il mouse ci è sopra

**Volume:**

Classe che gestisce l'impostazione del volume dell'applicazione

Attributi:

-**private float standard:** questo attributo serve a memorizzare il valore predefinito del volume dell'applicazione. Questo valore rappresenta il livello di volume standard che l'applicazione utilizzerà come riferimento per il volume iniziale o quando viene effettuato il reset del volume.

-**private float vol:** questo attributorappresenta il valore attuale del volume dell'applicazione. Viene utilizzato per tenere traccia del volume corrente

-**private AudioPlayer output:** questo attributocontrolla l'audio dell'applicazione. Questo oggetto viene utilizzato per regolare il volume dell'audio

Metodi:

-**public Volume(AudioPlayer output, float s):**questo metodo è il costruttore

-**public void more():**questo metodo serve per l’incremento del volume

-**public void less():**questo metodo serve per il decremento del volume

-**public void reset():**Metodo che resetta il volume al valore predefinito

-**public void standard(float s):**Metodo che imposta il volume predefinito

**Package Wifi:**

**WifiConnection:**

Classe per la comunicazione tramite WiFi

Attributi:

-**private WebsocketServer ws:** rappresenta un'istanza, che è utilizzata per gestire il server WebSocket.

-**private int now:** rappresenta il timestamp dell'istante attuale ottenuto utilizzando il metodo millis() dell'oggetto PApplet.

-**private PApplet a:**questo attributo permette l’utilizzo dei metodi della libreria Processing.core.\*

Metodi:

-**public WifiConnection(PApplet a):** questo metodo è il costruttore

-**public void write(Object c):** metodo che serve a inviare un messaggio tramite la connessione WebSocket

-**public void webSocketServerEvent(String msg):** metodo per la ricezione dei messaggi inviati dal server

Librerie utilizzate

-**importazione della libreria di processing:** processing.core.\*

Scopo d’utilizzo:

Per l’utilizzo delle funzionalità di processing

-**importazione della libreria BluetoothScreen:** bluetoothScreen.KeyStrokes.\*

Scopo d’utilizzo:

Per l’utilizzo della funzionalità della classi KeyStrokes

-**importazione della libreria Controlp5:**Controlp5.\*

Scopo d’utilizzo:

Per l’utilizzo della creazione dei bottoni

-**importazione della libreria java.util.\*:** java.util.\*

Scopo d’utilizzo:

Per l’utilizzo di tutti le funzionalità delle liste

-**importazione della libreria ddf.minim.Minim:** ddf.minim.Minim.\*

Scopo d’utilizzo:

Per l’utilizzo della musica e dei suoni

**Specifichiamo che l’uso del WI-FI non è stato più Implementato a causa di problema Hardware.**